DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

AN INTERACTIVE COMPUTER AID
FOR THE DESIGN AND ANALYSIS OF
LINEAR, SINGLE INPUT/SINGLE OUTPUT
DIGITAL AND CONTINUOUS CONTROL SYSTEMS

by

Clifton M. Cooksey

December 1984

Thesis Advisor: Marle D. Hewett

Approved for public release; distribution unlimited.

T222027

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) An Interactive Computer Aid for the Design and Analysis of Linear, Single Input/Single Output Digital and Continuous Control Systems		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis December 1984 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Clifton M. Cooksey		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93943		12. REPORT DATE December 1984 13. NUMBER OF PAGES 129
14. MONITORING AGENCY NAME & ADDRESS(If different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Classical Control Analysis
Root Locus Analysis
Frequency Response Analysis
Time Response Analysis
Digital Control System
s-plane
z-plane
w-plane
w'-plane

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

In this thesis the development of the Digital and Analog Control System Analysis Program (DACSAP) is discussed. DACSAP is an interactive computer aid for the design and analysis of linear, single input/single output feedback control systems. DACSAP is user friendly; it uses menus for option selection and prompted data entry. The program will analyze systems which are described by transfer functions written in the s, z, w or w' domains. The

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT continued

program will manipulate the transfer functions of multi-loop systems to produce the open and closed loop transfer functions required for a variety of analysis techniques. The analysis techniques included in DACSAP are root locus, open and closed loop Bode frequency response, Nyquist frequency response, Nichols frequency response and closed loop time response. The output of any of these analysis techniques may be either a tabulation of data points or a high resolution plot.

Approved for public release; distribution is unlimited.

An Interactive Computer Aid for the Design and Analysis of Linear, Single Input / Single Output Digital and Continuous Control Systems

by

Clifton M. Cooksey
Lieutenant, United States Navy
B.S., United States Naval Academy, 1977

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
December 1984

74782

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

ABSTRACT

In this thesis the development of the Digital and Analog Control System Analysis Program (DACSAP) is discussed. DACSAP is an interactive computer aid for the design and analysis of linear, single input / single output feedback control systems. DACSAP is user friendly; it uses menus for option selection and prompted data entry. The program will analyze systems which are described by transfer functions written in the s, z, w or w' domains. The program will manipulate the transfer functions of multi-loop systems to produce the open and closed loop transfer functions required for a variety of analysis techniques. The analysis techniques included in DACSAP are root locus, open and closed loop Bode frequency response, Nyquist frequency response, Nichols frequency response and closed loop time response. The output of any of these analysis techniques may be either a tabulation of data points or a high resolution plot.

DUDLEY KNOX LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA 93943

TABLE OF CONTENTS

LIST OF FIGURES

SYMBOLS

$D()$ = denominator polynomial of the open loop transfer function

$G()$ = forward path transfer function

$GH()$ = open loop transfer function

$H()$ = feedback path transfer function

K = open loop gain

$N()$ = numerator polynomial of the open loop transfer function

s = Laplace transform operator

$T()$ = closed loop transfer function

w = bilinear w-transform operator

w' = bilinear w'-transform operator

$x(z)$ = z-transform of $x^*(t)$

$x^*(t)$ = discrete output signal

z = discrete z-transform operator

δ = input signal

ϕ = phase of $GH(s)$ evaluated at $j\omega$

ν = transformed frequency in w-plane

ν' = transformed frequency in w'-plane

ω = real input frequency

ω_n = undamped natural frequency

ζ = damping ratio

ACKNOWLEDGEMENT

I wish to dedicate this thesis to my wife, Patricia. Without her constant love, support and understanding this work would not have been possible.

I. INTRODUCTION

The analysis of control systems, especially large systems, involves a great deal of calculations. In the case of the classical analysis of single input / single output systems, these calculations involve the manipulation of transfer functions. In the design process, where the characteristics of the system components may change many times, the repetitious recalculation of transfer functions can be very time consuming. Additionally, classical analysis techniques are, generally, graphical in nature. The drawing and redrawing of plots for analysis can also take up a considerable amount of the designer's time. The use of a computer to manipulate transfer functions and draw plots would allow the designer to spend a greater amount of his time designing.

A. BACKGROUND

The development of a computer aid for control system analysis is certainly not a new idea. Many such aids already exist. However, most of these programs are specialized in a particular type of analysis technique and many are not interactive. Thus, the designer must typically implement several of these programs during the design process. Each of these programs may have different methods for inputting data, different methods for controlling the program and different types of output. Several problems which exist with these programs are listed below. An attempt was made to minimize these weaknesses during the development of DACSAP.

1) The programs are not interactive. In these cases the user must, normally, type data into a data file using a very specific format.

2) Non-interactive programs will halt execution if data is not formatted properly or will run to completion producing meaningless output and will often leave the operator with no idea where the error was made.

3) Non-interactive programs do not offer the user any program control, thus, he cannot tailor the output to his needs. As a result, reams of tabular output may be produced when only a plot is desired or vice versa.

4) The data file for non-interactive programs must be re-accessed and edited and the program re-executed each time a change to the system is made.

5) Those programs which are interactive often require the user to know a specific language or set of commands in order to operate the program.

6) In either of these types of programs, the operator must have in his possession a user's manual which outlines the input format or the operating commands required to perform specific tasks.

7) The programs are specific to one type of analysis. The designer must, then, execute several programs to complete his work, retyping the input data and re-familiarizing himself with the operating instructions for each.

Although these programs may perform the required calculations, a great deal of the user's time is spent as a computer operator and less as a control system designer.

B. PROBLEM OBJECTIVE

The objective of this project was to design a comprehensive program which would meet the needs of the designer without the problems listed in the previous section. To meet this goal, the following requirements were used as a guideline during the development of the program:

1) The program should provide all of the most commonly used classical control system analysis techniques.

2) The program must be able to analyze both continuous and digital systems. Therefore, transfer functions in the s, z, w, w' domains must be handled.

3) The program must be interactive. The user should be able to change any of the system parameters at any time without retyping all parameters. He should also be able to go from one analysis technique to another without having to re-enter any of the system parameters.

4) The program should be easy to operate or be "user friendly". Errors input by the user should be easy to detect and correct. The user should have complete

control of the program, but not be required to know any special language or commands.

5) The output of the program should be selectable by the user and meet his immediate needs.

6) The program should take advantage of state-of-the-art technology using available computer power, memory and graphics capabilities.

The requirements listed above are fairly general in nature. There are several ways of meeting these requirements, particularly in the area of user friendly, interactive programming. The following chapter describes, in more detail, the methods used in meeting the objectives of this project.

II. DEVELOPMENT OF THE PROGRAM

To meet the objective outlined in the previous chapter, the Digital and Analog Control System Analysis Program, or DACSAP, was developed. DACSAP is an interactive program written in FORTRAN to run on the IBM 370 model 3033 computer at the Naval Postgraduate School. The FORTRAN programming language was chosen for the following reasons:

1) A large library of FORTRAN functions and subroutines exist to perform routine numerical calculations.
2) FORTRAN programs can interface with the state-of-the-art graphics package DISSPLA (see Appendix C).
3) Since FORTRAN is a common programming language, it will be easy for programmers to alter and improve DACSAP in the future.

A. APPROACH TO THE PROBLEM

A top-down programming technique was used to develop DACSAP. First, the tasks that the program was to perform were defined. Then, each task was developed in greater detail until, finally, the very specific routines required to perform each task were implemented. Of course, a few new tasks were defined and routines created during the development of the program and were incorporated in parallel with existing routines.

The tasks which the program performs may be broken into three basic categories:

1) Those involved in manipulating the system's transfer functions.
2) Those involved in performing analysis of the system.
3) Those involved in the control of the program.

The approach used in accomplishing each of these tasks is discussed in the following subsections.

1. System Manipulation

DACSAP uses block diagrams to depict control systems. Each block in the diagram represents a component in the system and is described by a transfer function.

Analysis is usually performed on the system's open or closed loop transfer functions. Therefore, DACSAP uses a single feedback control loop consisting of blocks in the forward and feedback paths for which the open and closed loop transfer functions may be calculated. Each block's transfer function is defined by a numerator and denominator polynomial in either coefficient or factored form. DACSAP makes available 20 blocks for transfer functions in the control loop. In most cases, this should allow the user to input the system exactly as it appears in the block diagram.

It is common for a single input / single output system to consist of several nested feedback loops. These systems can be reduced to single loops by calculating the closed loop transfer function of the inner loops and including them as separate blocks in the next outer loop. This method of block diagram manipulation is performed automatically by DACSAP and used to reduce multi-loop systems to a single loop for analysis.

Routines were developed to allow the user to input multi-loop systems as they appear in a block diagram by starting on the innermost loop and expanding to succeeding outer loops. DACSAP can be made to calculate the closed loop transfer function for the inner loop and place it anywhere in the next outer loop.

It was a primary objective to be able to change the system at any time with minimum effort. To this end, routines were developed to allow the user to change a single coefficient or root, redefine an entire polynomial, redefine an entire block or add new blocks. It was required that any

of these changes could be made without effecting any other part of the block diagram and that the resulting changes in the system's open and closed loop transfer functions be automatically recalculated.

2. Analysis Techniques

One of the major objectives of the project was to make available, in a single program, as many control system analysis techniques as possible. The following is a list of the analysis techniques chosen for inclusion in DACSAP:

1) Root locus analysis
2) Closed loop Bode frequency response analysis
3) Open loop Bode frequency response analysis
4) Nyquist analysis
5) Nichols analysis
6) Time response analysis

Each of these analysis techniques may be performed on digital or continuous systems. Thus, the user may choose to represent the system using s, z, w or w' transfer functions. In all of the analysis techniques either the method of calculation or the way the data is presented differs depending on which domain is being used. So, routines were developed to handle analysis in any domain. This was done in all but one case. Frequency response of digital systems must be done in either the w or w' domains.

It was also a requirement that the user be able to switch from one analysis technique to another without having to reenter any system parameters. This was easily accomplished by using common block variables and arrays. During the development of the analysis routines, it was also found useful to be able to retain analysis parameters between executions of a routine. For example, if a system is analyzed using Bode analysis, the user can leave the Bode routine, make changes to the system and rerun the Bode

analysis for the new system using the same frequency response parameters he had used before without having to reenter any of them. Of course, any or all of the parameters may be changed if the user so desires.

3. Program Control

Another of the project's objectives was to develop a user friendly method of program control which was flexible, but which did not require the user to know any special language or commands. This was accomplished using two concepts. First, the program is, to a certain extent, self managing. This concept is applied only in those areas where total user control is not felt to be necessary. For example, some programs require that the user instruct the program to calculate the open or closed loop transfer functions after the system has been entered and prior to commencing any analysis. If the system is changed, instructions must, again, be given to recalculate the transfer functions required for further analysis.

DACSAP's operation is such that transfer functions are automatically calculated as they are needed to perform analysis calculations. These transfer functions are also automatically updated when any changes are made to the system by the user. Although some control of the program is taken away from the user in these cases, the chance of making an error is also reduced and the user's time is spent more efficiently.

The second concept used in program control is the use of option menus. A variety of menus are presented to the user throughout the program's execution showing the options available to him at that time. Self-management is also sometimes used in the control of the option menus. That is, the user is rarely given the opportunity to select which menu he will see next. Rather, the correct menu is

automatically presented to the user based on where execution is occurring in the program and which tasks have been completed to that point.

The result of menu driven options is that the options available are presented in plain English so that the user does not have to know any special language or commands. Since the control of the menus is handled by the program, the user is always presented with a menu of options when one is required and that menu only contains options which are of interest to him at that time. These concepts allow the program execution to flow with a minimum number of inputs and far fewer errors.

B. ORGANIZATION OF THE PROGRAM

As mentioned in the previous section, the top-down approach was used to design DACSAP. Subsequent to the design, specific task requirements were defined. The routines to perform these specific tasks were written first. Thus, that the program was actually written from the bottom up.

Low level routines were written based on their required output. The input requirements of these routines, then, defined the output of the next higher routine. This process of building upward was continued until the required data was that data which must be supplied by the user. For this reason, the parameter input routines form the highest levels of the program.

The basic organization of the program is shown in Figure 2.1. The lowest level routines are the analysis routines; these were written first. The calculation routines were written next and the parameter input routines last. The parameter input routines take inputs from the user and put them into forms which may be used by the calculation

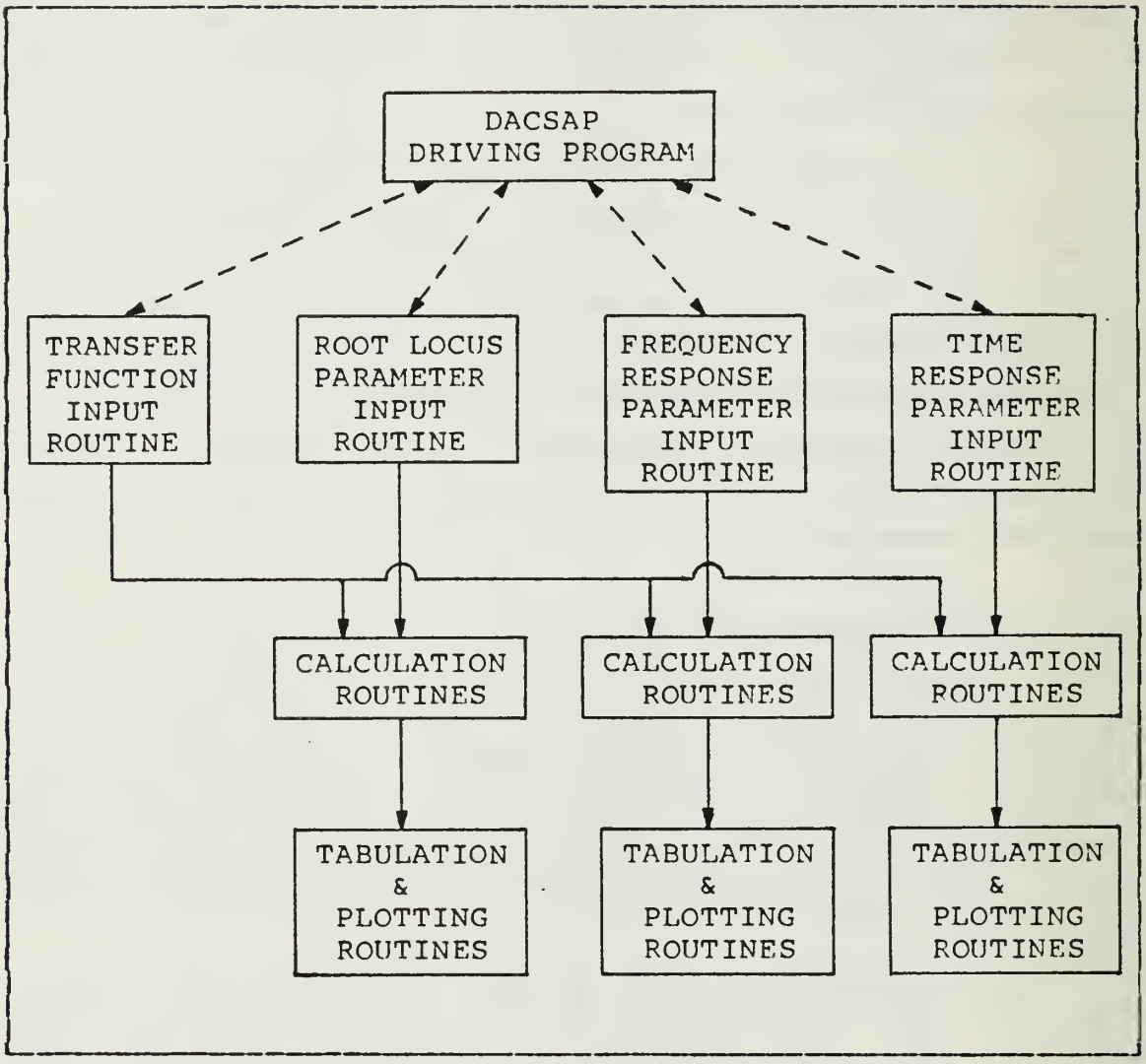


Figure 2.1 Program Organization.

routines. The user controls the program primarily through these parameter input routines. Thus, these routines contain the instructions for presenting option menus and for changing and correcting parameters.

The uppermost controlling routine is the driving routine, called DACSAP, which provides the main option menu. This routine shifts program control to the parameter input routines based on the selection from the main menu. The

dashed lines in Figure 2.1 show how control is shifted. Once the required tasks have been performed by the analysis routines, control is shifted back to the driving program. The shifting of control to the calculation and output routines is done automatically and is not controlled by the user.

The flow of information is depicted by the solid lines in Figure 2.1. The user supplies system data via the transfer function input routine and analysis parameters via the root locus, frequency response and time response parameter input routines. Information is taken from these routines by the calculation routines which, in turn, provide information to the tabulation or plotting routines.

The following chapter contains a more detailed description of each of the branches shown in Figure 2.1.

III. PROGRAM DESCRIPTION

DACSAP consists of a small driving program and 40 subroutines. The driving program, which is called DACSAP, does little more than present the user with the main option menu (Fig 3.1) and call the appropriate subroutines based on the selection made from that menu.

DACSAP CPTIONS	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	BODE ANALYSIS (OPEN LOOP)
4	BODE ANALYSIS (CLOSED LOOP)
5	NYQUIST ANALYSIS
6	NICHOLS ANALYSIS
7	TIME RESPONSE
8	CHANGE BLOCK DIAGRAM
9	SAVE THIS PROBLEM
10	START A NEW PROBLEM
11	HELP
12	EXIT DACSAP

Figure 3.1 DACSAP's Main Option Menu.

All calculations, input/output functions and remaining program control functions are performed by DACSAP's subroutines. These subroutines fall into one of the following four basic categories:

1) parameter input routines
2) calculation and analysis routines
3) plotting or tabulation (output) routines
4) utility routines.

The four main branches of the program (transfer function input and three analysis sequences) all contain subroutines from each of these categories, except for the transfer function input sequence which has no output routine. Each of the four branches will be described in greater detail later in this chapter.

First, however, it is necessary to take a closer look at the control structure of the branches. Program self-management is used to a greater extent when an analysis routine is first entered. Figure 3.2 depicts the relation-

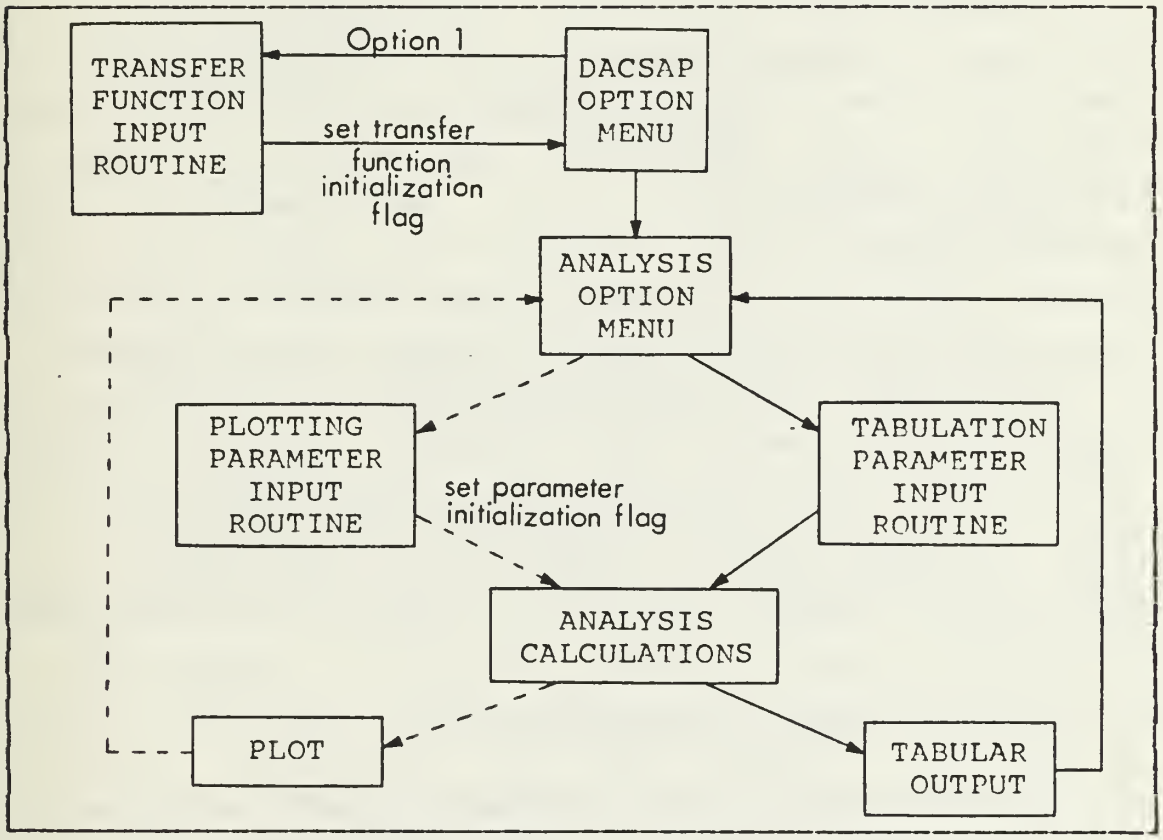


Figure 3.2 Control Structure Before Parameter Initialization.

ship between the main menu (DACSAP), the transfer function input routine and the analysis routines. Before analysis

can be performed, the system must be defined by using Option 1 of the main menu. Once the system has been input, the transfer function initialization flag is set. The significance of this is explained later.

When an analysis technique is chosen from the main menu, the user is presented with the option menu for that branch (root locus, frequency response or time response). The user may select graphical or tabular output which will send him to the appropriate parameter input routine. Once all of these parameters have been entered, another flag is set. Then, analysis calculations and output automatically occur.

After the first run through the analysis routines, the structure of control changes to that depicted in Figure 3.3. The change from the first structure (Fig 3.2) to this one is dependent on the condition of the transfer function and parameter initialization flags. Note that more direct control of the program is given to the user via the change menus. This allows the user to make selected changes to any parameter and rerun the analysis in any sequence desired.

The following sections briefly describe the function and capabilities of the four main branches. The basic mathematical concepts used will be stated but not carried out in any detail. The reader is referred to the bibliography of this report for a list of texts which cover control system theory in detail.

A. TRANSFER FUNCTION MANIPULATIONS

Transfer function manipulations are performed by several subroutines which permit the user to interactively input, change, store or load a system's transfer functions.

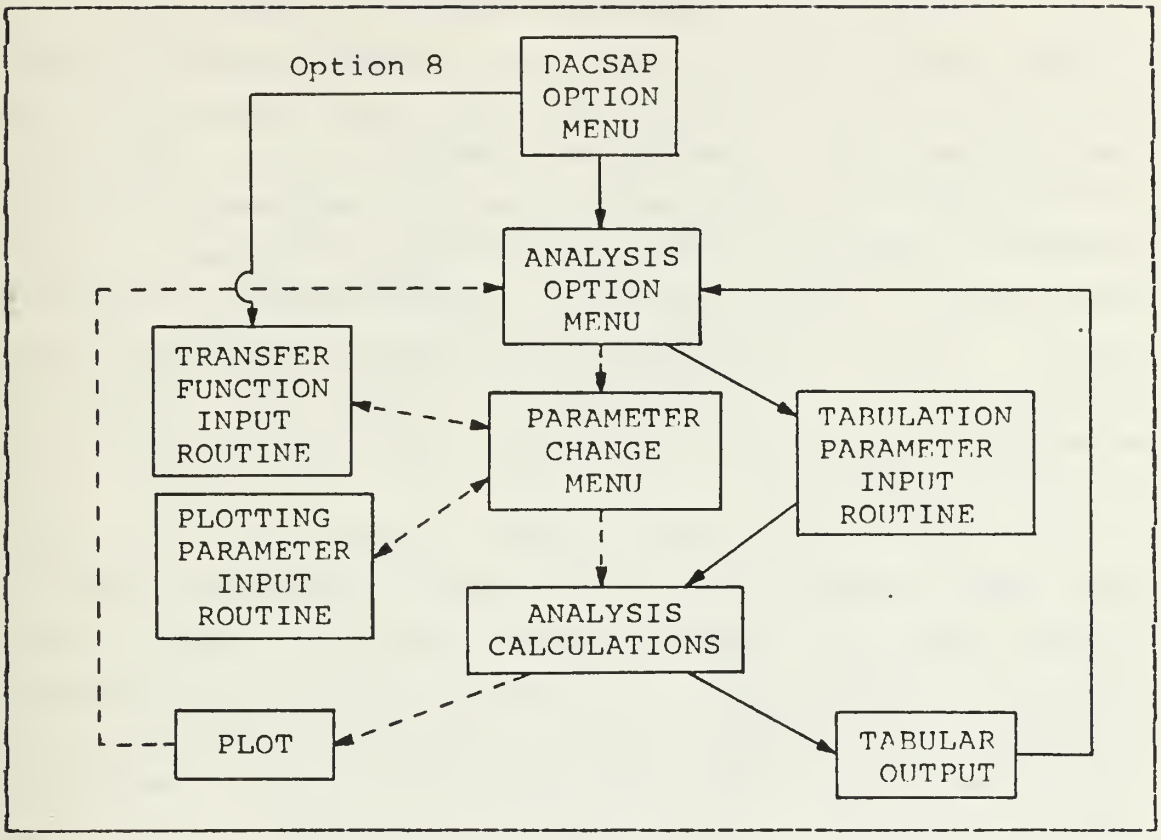


Figure 3.3 Control Structure After Parameter Initialization.

1. Transfer Function Input Routines

The purpose of the input routine, TINPUT, is to permit the user to input a system's transfer functions just as they appear in a block diagram, then calculate the system's open loop transfer function for use by the analysis routines. Because of the large number of calculations required for transfer function manipulation, TINPUT is the largest of DACSAP's subroutines.

The first time that the routine is entered, the user is given the opportunity to input the transfer functions from the console or from a data file. Entry from the console will be discussed first.

The system may consist of up to 20 components in the forward and feedback paths of a single feedback control loop. Each component must be defined by a transfer function and by the path (forward or feedback) in which it is located. The numerator and denominator polynomials for each transfer function are entered separately and may be described by either polynomial coefficients or polynomial roots. After each polynomial has been entered, the user is presented with the coefficients or roots just entered and given the opportunity to change any which may have been entered incorrectly.

Instead of typing in the transfer function polynomials for a given block, the user may choose to load the block's transfer function from a data file (creating data files is discussed later in this section). If this option is chosen, the numerator and denominator polynomial coefficients are pulled from the disk file, via the subroutine TLOAD, and placed in the appropriate arrays for the current block. The purpose of this feature is to allow one to design subsystems separately and store their transfer functions to disk. Then, when designing the outer loops of the system, the subsystem may be loaded directly into its proper block.

As previously mentioned, when the transfer function input routine is first entered, the user may choose to load the entire system from a data file. If this option is chosen, all data pertaining to a stored control loop will be loaded from a user specified data file. This feature allows the user to continue the analysis or design of a system that had been worked on during a previous terminal session without having to retype any system parameters. When the load is completed, the program will continue its operation as if the user had just finished entering the transfer functions from the console.

Once all transfer functions have been entered, either from the console or a data file, and the user is satisfied that they are correct, TINPUT calculates the steady state gain, roots and coefficients of the system's open loop transfer function. The results of these calculations are stored in common variables and arrays to be used by the analysis routines.

2. Transfer Function Change Routine

The instructions required to change or expand the block diagram are also contained in TINPUT. When any changes are made to the system, the routine automatically recalculates the parameters of the open loop transfer function. The option to change transfer functions may be selected from DACSAP's main menu or from any of the analysis routines' change menus. Those changes which may be performed on the block diagram are shown in Figure 3.4.

TRANSFER FUNCTION (T.F.) CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE A T.F. IN THE CURRENT LOOP
2	ADD A NEW BLOCK TO THE CURRENT LOOP
3	CHANGE T.F. IN AN INNER LOOP
4	EXPAND TO AN OUTER LOOP
5	NO CHANGES
6	HELP

Figure 3.4 Transfer Function Change Menu.

If the first option is chosen, the user will be asked which block he wishes to change and is presented with the block change option shown in Figure 3.5.

BLOCK CHANGE OPTIONS BLOCK NUMBER 1	
OPTION NO.	OPTION
1	CHANGE CURRENT NUMERATOR
2	CREATE A NEW NUMERATOR
3	CHANGE CURRENT DENOMINATOR
4	CREATE A NEW DENOMINATOR
5	CREATE NEW NUMERATOR & DENOMINATOR
6	MOVE BLOCK TO THE FEEDBACK PATH
7	NO MORE CHANGES TO THIS BLOCK

Figure 3.5 **Block Change Menu.**

Changing the block's numerator or denominator (options 1 or 3) means that the user simply wishes to alter one or more of the polynomial's coefficients or roots leaving all other block parameters unchanged.

By creating a new numerator or denominator (options 2 or 4), the user will be able to change the order of the polynomial and must enter all of the roots or coefficients for that polynomial. The form of the polynomial (factored or coefficient) must, however, remain the same as previously selected.

By selecting the option to create a new numerator and denominator (option 5), all data concerning the current block is, essentially, erased and the user is sent to the transfer function input routine to re-enter a transfer function for that block. This permits one to enter transfer function polynomials of different form and order into any block of the control loop. Just as one may initially enter transfer functions from a file, the user may choose to create the new numerator and denominator by loading them from a data file.

One may move a block from the forward path to the feedback path and vice versa (option 6). All other features about the block, including the block number, remain unchanged. Only its location within the control loop is changed.

If the second option from Figure 3.4 is chosen, the user is simply sent to the transfer function input routine to enter data for a new block. The block may be added to the forward or feedback path and may be entered from the console or from a data file.

DACSAP was designed to operate on a single feedback control loop. However, since control loop transfer functions may be saved to and loaded from data files, multiple loop systems may be designed and analyzed in the following manner.

The user designs the inner loop(s) of the system first, such as Loop 1 in Figure 3.6. Once the designer is

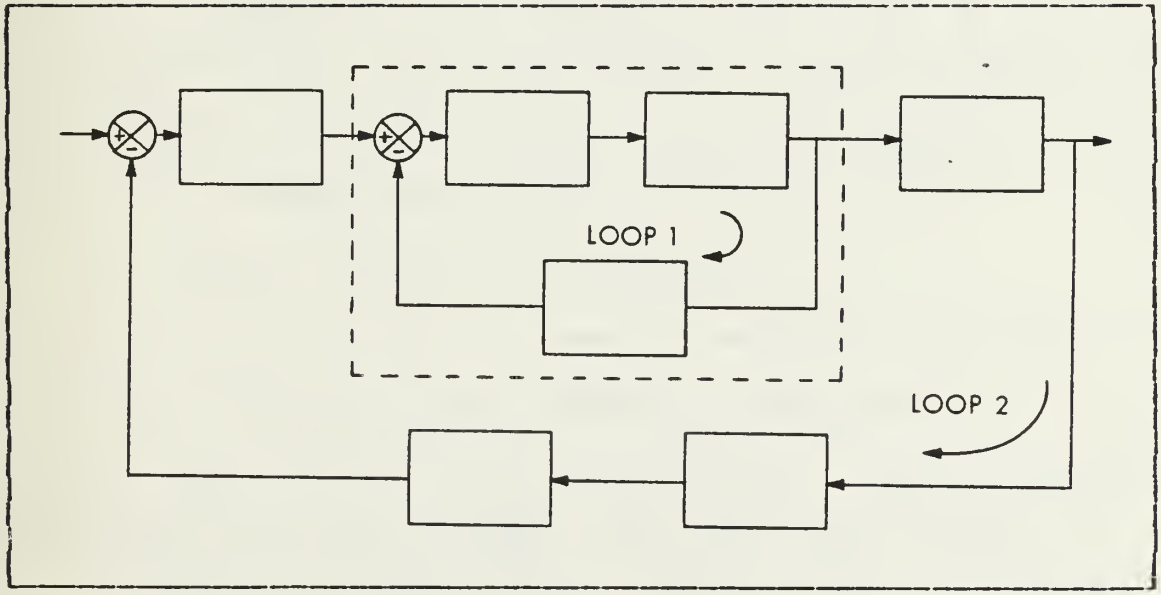


Figure 3.6 Sample Multiloop System.

satisfied that Loop 1 meets his specifications, he saves the system parameters and starts a new problem by loading the inner loop's closed loop transfer function into an outer loop block and entering the remaining blocks as he wishes. Option 4 of Figure 3.4 simplifies this process.

In option 4, those procedures are performed exactly as stated in the last paragraph except that the proper sequence of events occurs automatically and the user is prompted for each step throughout the procedure. The user may choose to place the inner loop transfer function into any block in the next loop. In the case shown in Figure 3.6, Block 2 is selected and the outer loop appears as shown

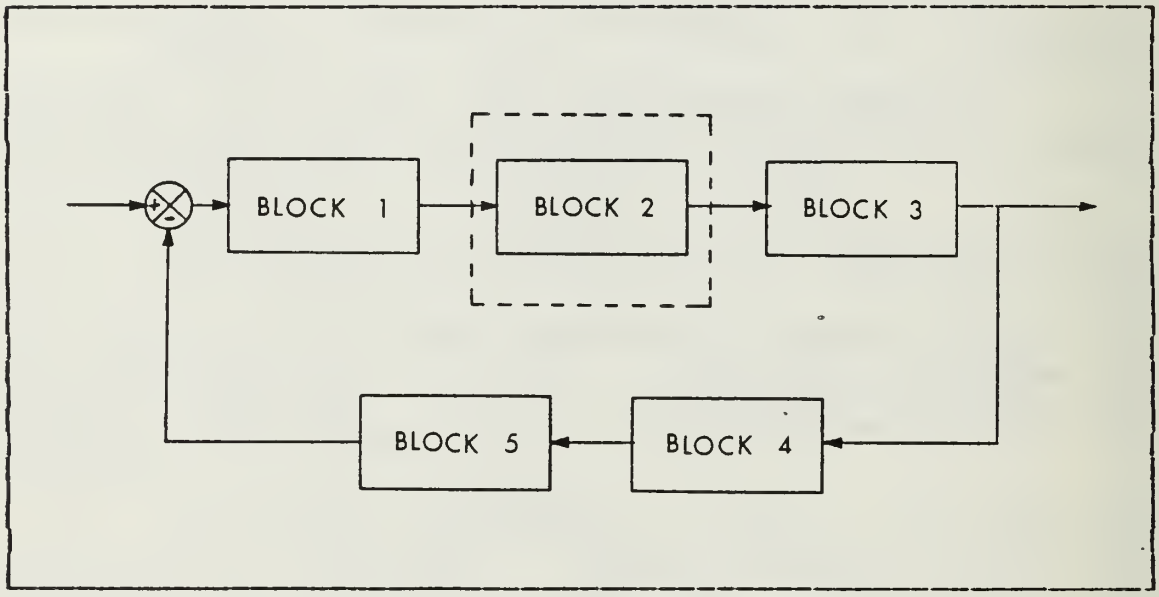


Figure 3.7 Reduced Multiloop System.

in Figure 3.7. An important feature of this option is that the program keeps track of the block in the outer loop containing the inner loop transfer function. This becomes important when this option (Option 4) is used in conjunction with Option 3.

In order to make changes in an inner loop, the user saves his outer loop, starts a new problem, loads the inner loop and makes the desired changes. Like Option 4, Option 3 (Fig 3.4) automatically controls this sequence of events, prompting the user each step of the way. After this option has been chosen and all changes have been made to the inner loop, the user may return to the outer loop which now reflects the changes made in the inner loop. This is done by selecting Option 4 to re-expand to the outer loop. The user is asked if the outer loop should be entered from the console or a data file. Since the outer loop was saved under Option 3, its data file can now be loaded. In this case, however, since the program has kept track of the block into which the inner loop was previously placed, it automatically updates that block with the new closed loop transfer function from the inner loop.

One can see that large multi-loop systems may be designed by creative use of Options 3 and 4 of the Transfer Function Change Menu. If each loop of the system is saved as it is expanded, the user is free to return to any of the inner loops, make changes and have those changes automatically incorporated in the outer loops by successively using Option 4 to re-expand the system.

3. Transfer Function Save Routine

The purpose of the save routine is to store all transfer function data to a disk file under a user specified file name. This function is performed by the subroutine TSAVE. TSAVE is called either by the user, through DACSAP's main menu, or by the transfer function change routine.

The data file set up by TSAVE contains two sections; the block load section and the system load section. When TSAVE is executed, the user is asked if the closed loop transfer function should be calculated and saved. The

response to this question determines which transfer function, open or closed loop, will be stored in the block load section of the data file.

The user's decision to store one or the other is based on the type of system being saved. For example, if the designer uses open loop Bode analysis to design a cascade compensator, the open loop (cascade) transfer function is saved in the block load section. If the inner loop of a multi-loop feedback control system had just been designed, the closed loop transfer function is saved so that it can be loaded into the outer loop.

The system load section contains all data which describes the entire control loop. These parameters include the domain of the transfer functions, the number of blocks, individual block transfer function polynomials, the open loop coefficients and roots and several other parameters. The system load section is always the same regardless of what is stored in the block load section.

The resulting data file, then, contains all of the data necessary to completely reconstruct the control loop. The entire system may be recalled later for further analysis or just the open or closed loop transfer function may be recalled for insertion into a block of another control loop.

B. ROOT LOCUS ANALYSIS

The purpose of this routine is to calculate the complex roots of the system's characteristic polynomial over a user specified range of open loop gain values. The major subroutines involved in root locus analysis are DACRUT, which is the parameter input routine, and RTLCS, which calculates and plots pole, zero and root locations for the system. The parameters which must be input by the user include a range of open loop gain values, the type of feedback (positive or

negative), the sample period for discrete systems and the dimensions and heading for the plot.

From elementary control system theory it is known that the characteristic equation for a single feedback control loop is

$$1 + K\ N(s)/D(s) = 0 \qquad (3.1)$$

for negative feedback or

$$1 - K\ N(s)/D(s) = 0 \qquad (3.2)$$

for positive feedback, where

N(s) = numerator polynomial of the open loop transfer function

D(s) = denominator polynomial of the open loop transfer function

K = open loop gain

Equations 3.1 and 3.2 may be rewritten, respectively, as

$$D(s) + K\ N(s) = 0 \qquad (3.3)$$

$$D(s) - K\ N(s) = 0 \qquad (3.3)$$

RTLCS calculates an open loop gain increment based on the range of gain values and the plotting dimensions chosen by the user. This increment provides the greatest number of plotted points for each locus. For each gain value, K, RTLCS calculates the coefficients of the characteristic polynomial (Eqns 3.3 or 3.4), determines the roots of that polynomial by calling the IMSL routine ZPOLR, and plots the root locations.

Lines of constant damping ratio and natural frequency are drawn on the root locus plot depending on which domain is being used. In the s-plane, lines of constant damping ratio are simply straight lines radiating from the origin and lines of constant natural frequency are circles about the origin. Because of this simplicity, s-plane plots are not cluttered by drawing these lines.

On the w-plane, lines of constant damping ratio are curved lines radiating from the origin and lines of constant natural frequency are ellipses about the origin. The w'-plane varies from one which looks like the w-plane to one similar to the s-plane depending on the sampling rate. Just enough lines are drawn on these plots to make the user aware of the behavior of damping ratio and natural frequency for the system. The w'-plane root locus plot in Figure 3.8 shows the lines of constant damping ratio and natural frequency.

On the z-plane, the stable region is within the unit circle and the lines of constant damping ratio and natural frequency follow a somewhat more complex pattern. The values associated with these lines vary with sample period. These complexities make analyzing z-plane root locus plots more difficult. To make the task easier, the stable region and lines of constant natural frequency and damping ratio are drawn and labelled in detail for z-plane plots (Figure 3.9).

After the root locus plot has been drawn, the user may request detailed information about specific root locations. The program returns values for open loop gain, natural frequency and damping ratio for these root locations. The user may, then, highlight the roots associated with a particular value of open loop gain.

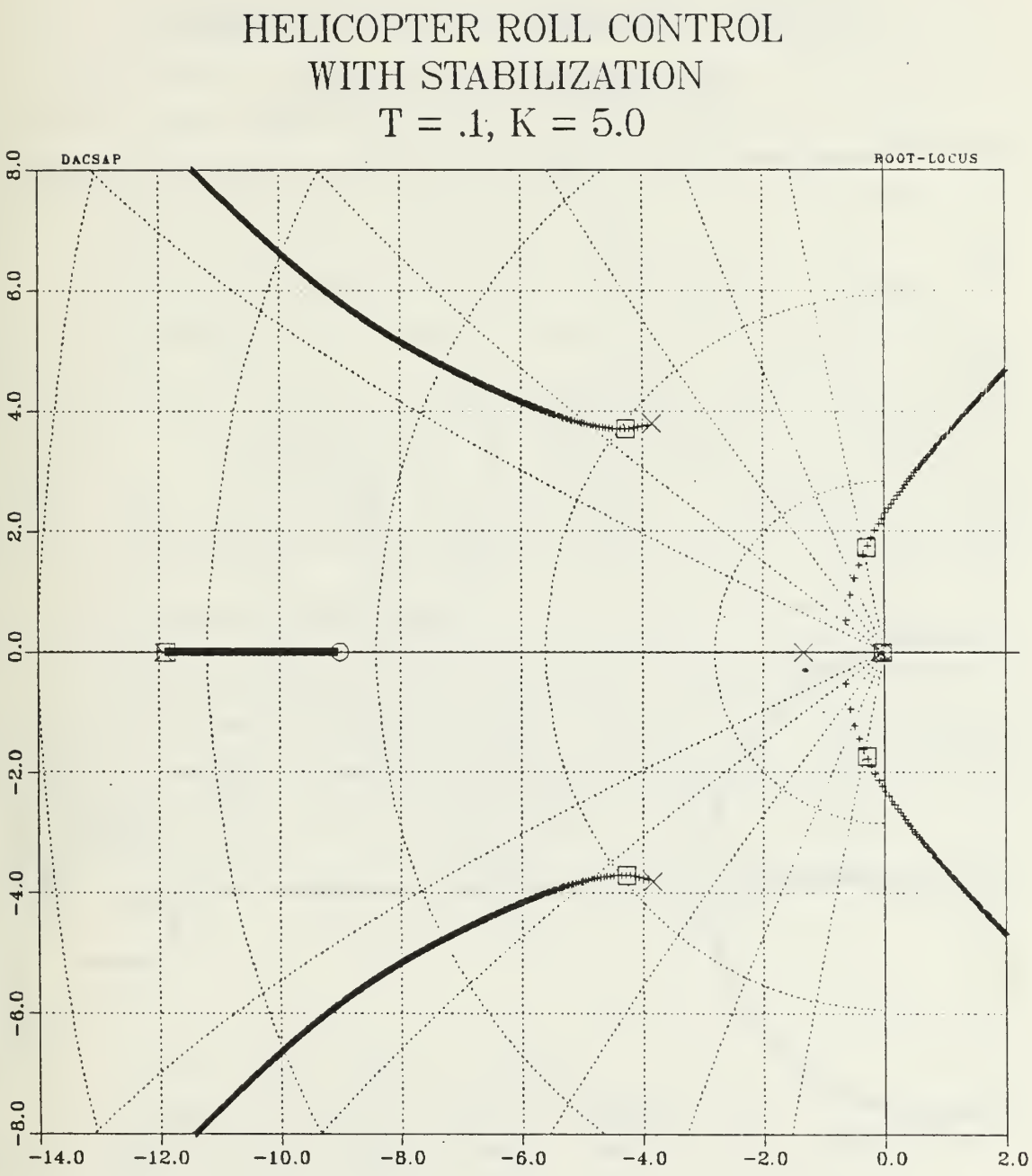


Figure 3.8 w'-plane Root Locus Plot.

DIGITAL ANTENNA AZIMUTH CONTROL

COMPENSATED SYSTEM

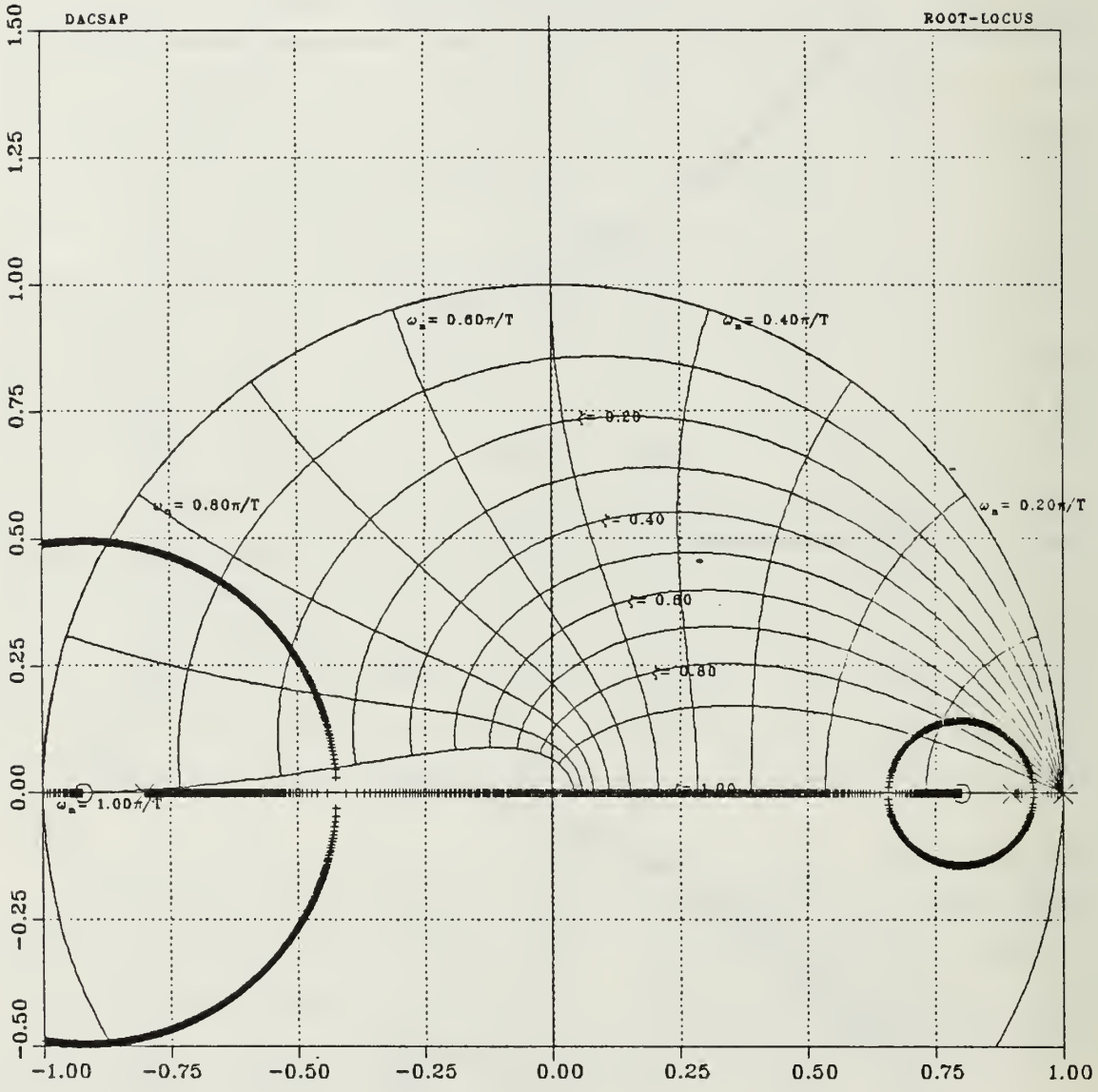


Figure 3.9 z-plane Root Locus Plot.

C. FREQUENCY RESPONSE ANALYSIS

These routines determine the frequency response of the system's open or closed loop transfer function. The output of the frequency response routines may be displayed graphically or in tabular form. Relative stability analysis may be performed by examining the system's open loop frequency response using Bode plots, polar plots (Nyquist analysis) or log magnitude-phase plots (Nichols chart analysis). The total system's steady state response to a sinusoidal input may be analyzed using closed loop Bode plots.

The major routines involved in frequency response analysis are DACFRQ, the parameter input routine, and FREQR, the calculation routine. Three plotting routines, BPLOT, NYPLOT, and NICPLT, are used to graphically display the response. In the following subsections the input parameters which must be provided by the user are discussed and the calculations performed by the program are briefly described.

1. Open Loop Analysis

The system's open loop frequency response is obtained by substituting $j\omega$ for s in the open loop transfer function, GH(s) (in the discrete domains, w is replaced by $j\nu$ or w' is replaced by $j\nu'$). The value of GH(s) at $j\omega$ is

$$GH(j\omega) = R(\omega) + j\, X(\omega) \qquad (3.5)$$

where

$R(\omega)$ = real part of $GH(j\omega)$

$X(\omega)$ = imaginary part of $GH(j\omega)$

GH(s) may also be expressed as

$$GH(j\omega) = |GH(j\omega)| \exp\{j\phi(j\omega)\} \qquad (3.6)$$

where

$$|GH(j\omega)| = \{R^2(\omega) + X^2(\omega)\}^{1/2} \qquad (3.7)$$

and

$$\phi(\omega) = \tan^{-1}\{X(\omega)/R(\omega)\} \qquad (3.8)$$

are the magnitude and phase, respectively, of $GH(j\omega)$.

The coefficients of the open loop transfer function are known from TINPUT. The user need only input, via DACFRQ, the open loop gain and the range of frequencies, ω, over which the response is to be observed.

For discrete systems, ν or ν' are sometimes called "fictitious" frequencies and are not the same as real frequency, ω. Instead,

$$\nu = \tan(\omega T/2)$$
$$\nu' = (2/T)\tan(\omega T/2)$$

If the user is working in a discrete domain, he may have the response represented as a function of either real or fictitious frequency. If real frequencies are desired, the user must supply the sampling period, T; the program will then perform the frequency transformation.

Once all of the above data has been input, FREQR evaluates $GH(j\omega)$ at values of ω within the frequency range selected by the user. Values of $R(\omega)$, $X(\omega)$, $|GH(\omega)|$ and $\phi(\omega)$ are calculated and placed in common arrays for use by the plotting or tabulation routines.

Examples of Bode, polar and magnitude-phase plots are shown in Figures 3.10, 3.11 and 3.12, respectively.

TACTICAL AIRCRAFT YAW DAMPER

K = .3218

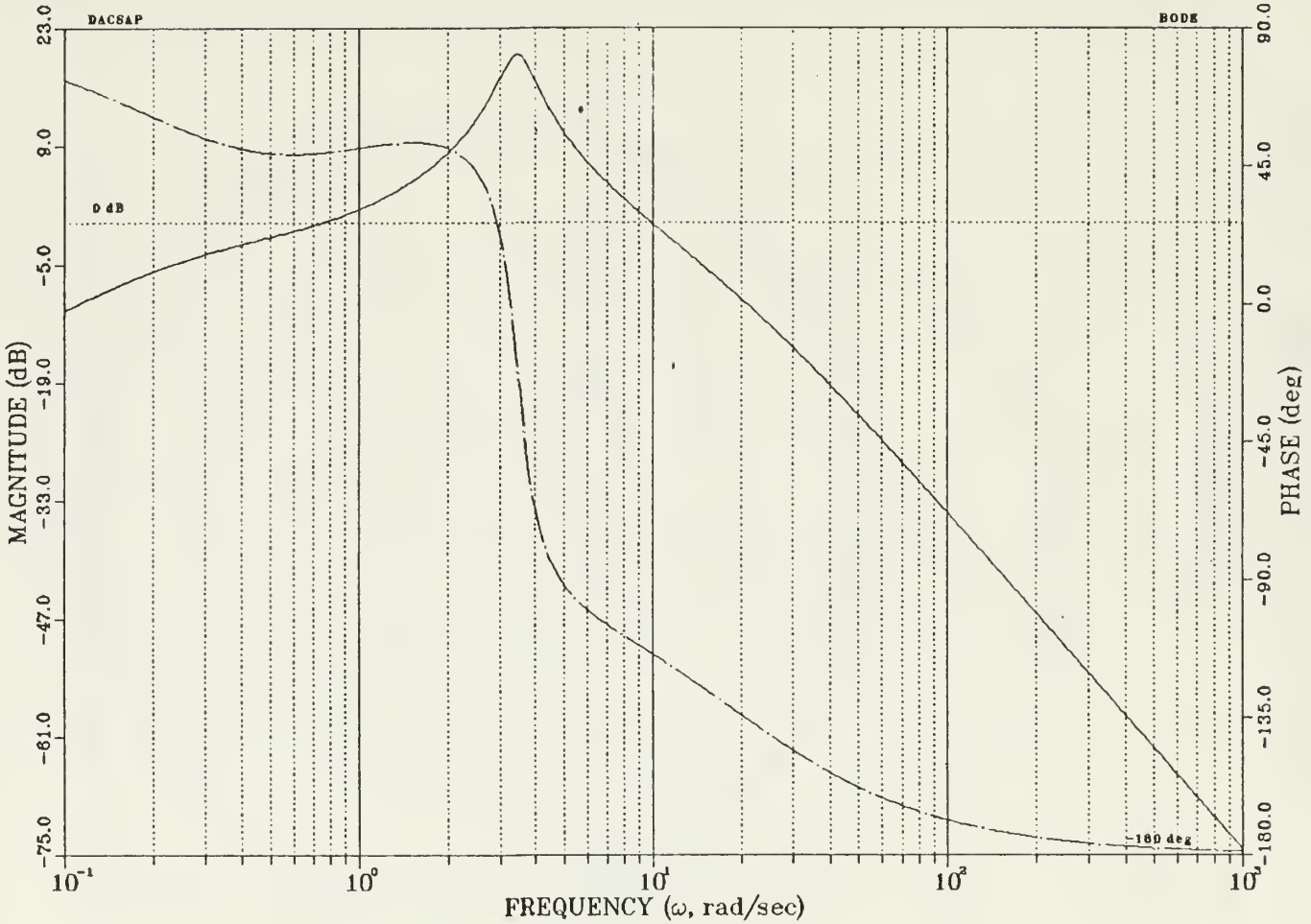


Figure 3.10 Bode Plot.

TACTICAL AIRCRAFT YAW DAMPER

K = .3218

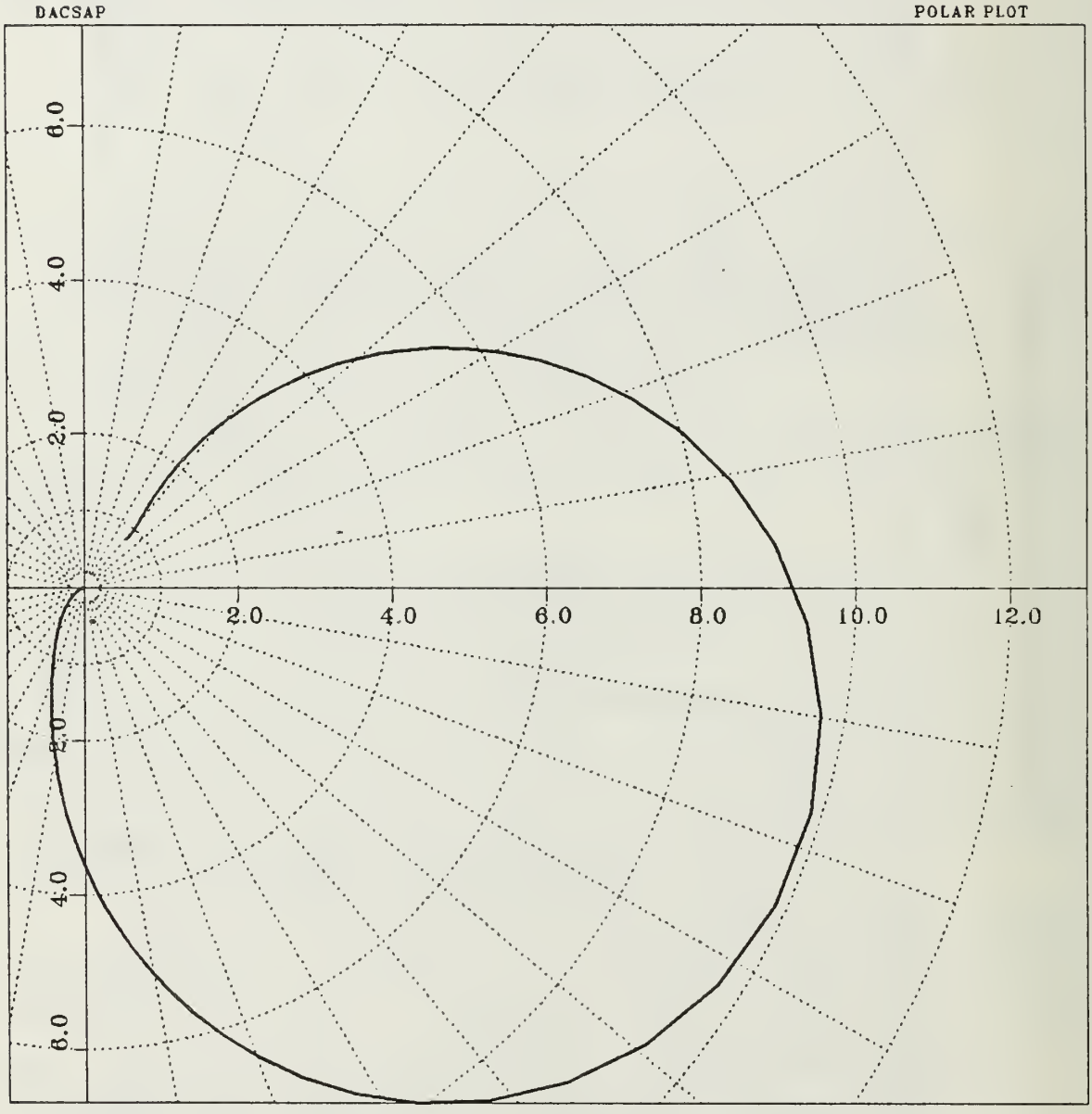


Figure 3.11 Polar Plot.

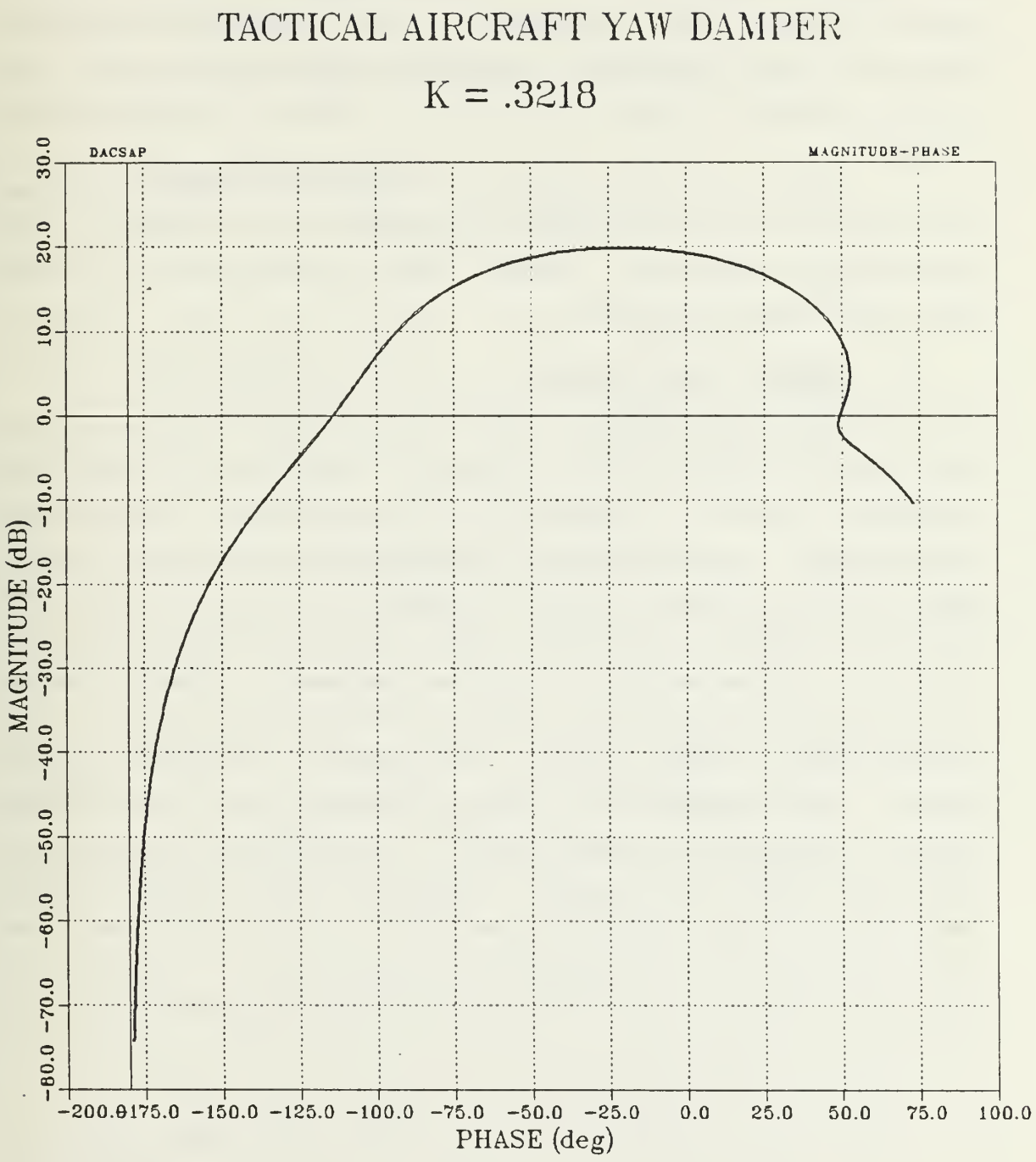


Figure 3.12 Log Magnitude-Phase Plot.

Note that the Bode and polar plots are complete and that Bode and Nyquist analysis and design techniques may be applied directly to these plots, either in hardcopy or on the graphics monitor. The magnitude-phase plot, however, must be transferred to hardcopy and an appropriately sized Nichols chart (template) must be available to the user before Nichols chart analysis or design techniques are employed.

For all of the techniques described above phase and gain margins and crossover frequencies are displayed when plotting is complete, if they can be determined over the specified frequency range.

2. Closed Loop Analysis

The calculations involved in closed loop frequency response analysis are very similar to those for open loop frequency response analysis, as discussed in the previous subsection. In this case, however, s is replaced by $j\omega$ in the system's closed loop transfer function. The input routine, DACFRQ, prompts the user for the forward path gain, feedback path gain and the type of feedback. These parameters are then used by the subroutine CLOOP to calculate the coefficients of the closed loop transfer function polynomials. The remainder of the parameters input via DACFRQ remain the same as described in the previous subsection. FREQR performs the frequency response calculations based on the closed loop transfer function and the user specified frequency range. Graphical presentation of the closed loop frequency response is in the form of a Bode plot.

D. TIME RESPONSE ANALYSIS

The routines described in this section provide the user with a time history of a closed loop system's response to a

user specified input. The user may choose to excite the system with either a unit impulse, a step function or a ramp function. The method for calculating a continuous system's response is different than that for a sampled (discrete) system. In either case, the user may display the response in tabular form or graphically as a plot of output magnitude versus time.

1. Continuous Systems

The major routines involved in evaluating continuous system time response are DACTIM, the parameter input routine, STIME and DVERK, the calculation routines, and TPLOT, the plotting routine.

The parameters which are input by DACTIM are the type of input (excitation), the closed loop parameters and the amount of time over which the response is to be observed. The closed loop parameters are values for the forward path, the feedback path gain and the type of feedback (positive or negative). The subroutine CLOOP calculates the coefficients of the closed loop transfer function based on these parameters and the open loop roots.

In order to calculate the continuous time response, it is desirable to perform the inverse Laplace transform for the closed loop transfer function, thus obtaining an expression for the output as a function of time. However, numerical methods for obtaining inverse Laplace transforms sometimes break down and cannot be relied upon in all cases. An alternate method is to generate a set of differential equations, and associated output equation, corresponding to the closed loop transfer function. A numerical method for solving that set of differential equations is then be employed and the output, as a function of time, is calculated.

From elementary linear control theory, a transfer function, T(s), can be changed to an equivalent set of state equations as follows. Given, for example, the transfer function

$$T(s) = \frac{s^3 + b_2 s^2 + b_1 s + b_0}{s^4 + a_3 s^3 + a_2 s^2 + a_1 s + a_0} \qquad (3.11)$$

the corresponding matrix equations are

$$\begin{vmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{vmatrix} = \begin{vmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ -a_0 & -a_1 & -a_2 & -a_3 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ 0 \\ 1 \end{vmatrix} \delta \qquad (3.12)$$

$$y = \begin{vmatrix} b_0 & b_1 & b_2 & 1 \end{vmatrix} \begin{vmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{vmatrix} \qquad (3.13)$$

where δ is the magnitude of the input at time, t. Equation 3.12 is the phase variable canonical form or companion matrix form of the state equation and is based on the denominator coefficients of the transfer function. Equation 3.13 is the output equation and is based on the zeroes' portion of the transfer function.

The subroutine STIME calculates the coefficients of the matrices in Equations 3.12 and 3.13. These coefficients are then used by the IMSL routine DVERK to create and solve the corresponding set of differential equations for values of time over the user specified time interval. Common arrays are established for the output amplitude and time

values and are made available for the plotting routine, TPLOT. Figure 3.13 is a sample continuous system time response plot.

2. Discrete Systems

As with continuous systems, it is desirable to obtain an expression, as a function of time, for the system's discrete transfer function. The inverse transform of a function written in the z-domain is obtained by examining the function's Taylor series expansion. By definition, the z-transform, $x(z)$, of a discrete signal, $x^*(t)$, is

$$x(z) = \sum_{k=0}^{\infty} x(kT)\ z^{-k} \qquad (3.14)$$
$$= x(0) + x(T)z^{-1} + x(2T)z^{-2} + \ldots + x(kT)z^{-k} + \ldots$$

In the case at hand,

$$x(z) = T(z)\ \delta(z) \qquad (3.15)$$

where $T(z)$ is the system's closed loop transfer function and $\delta(z)$ is the z-transform of the input signal. $T(z)$ is obtained from the routine CLOOP. ZTIME calculates $\delta(z)$ for the type of input specified by the user and combines it with $T(z)$ to form the rational function $x(z)$. The function $x(z)$ is expanded to an infinite power series by simple polynomial long division. For the convergent series, the coefficients of the z^{-k} terms are the values of $x(kT)$ in the pulse train $x^*(t)$.

TACTICAL AIRCRAFT YAW DAMPER

K = .3218

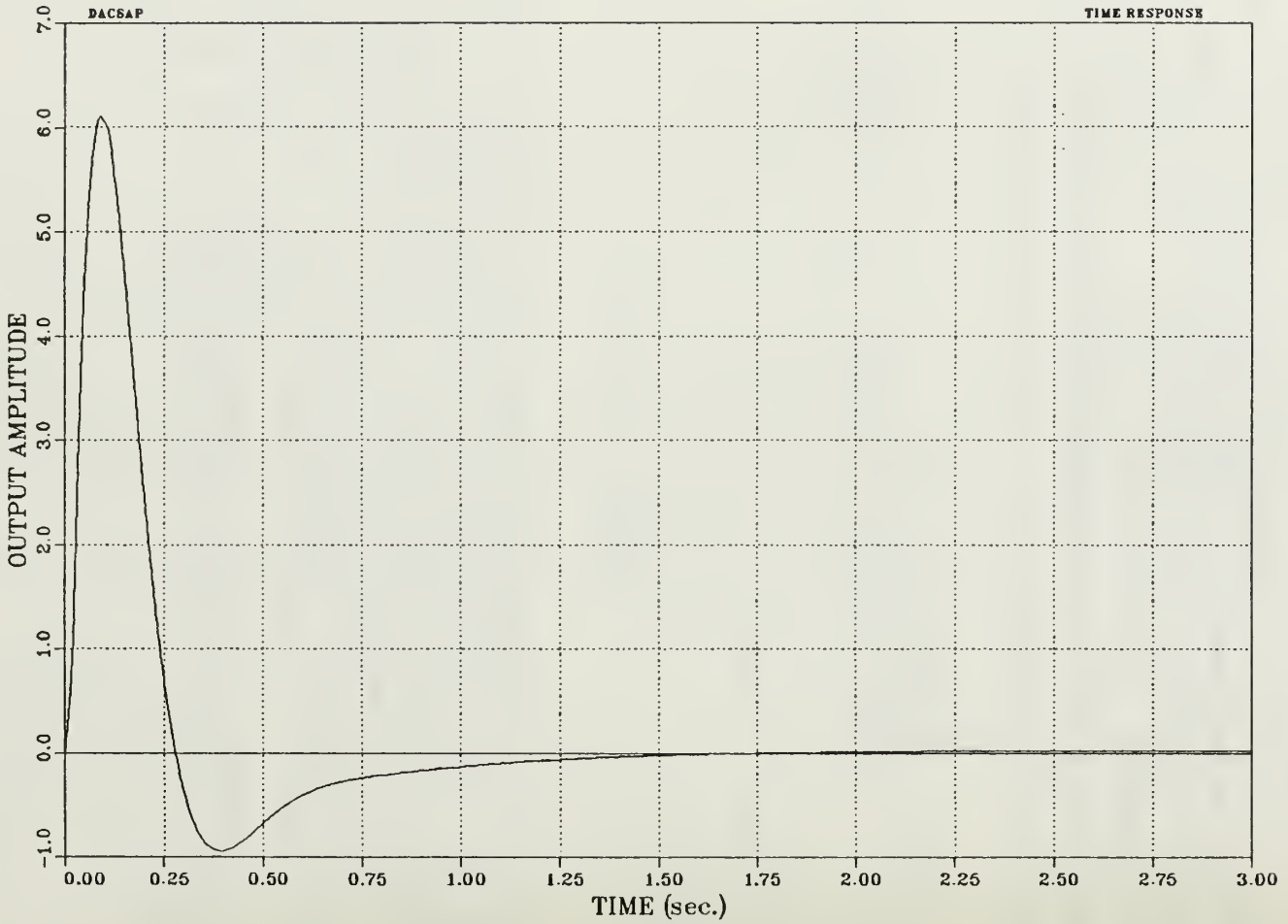


Figure 3.13 Continuous System Time Response Plot.

In the case of systems described by w- or w'-domain transfer functions, the same method is used. First, however, the routine WINV is used to perform the algebraic inverse transform from w to z or w' to z. Then the evaluation of the resulting z-domain transfer function is as described in the previous paragraph.

The outputs of ZTIME are common arrays containing the values of $x^*(t)$ and t for the sampling intervals, kT. Since $x^*(t)$ is not a continuous function, values are plotted by TPLOT as discrete, unconnected points as shown in Figure 3.14.

E. HELP ROUTINE

The word HELP appears as an option in the DACSAP main option menu, the transfer function change menu, and in each of the analysis routines' option menus. If this option is chosen from any of these menus, other than the DACSAP menu, several pages of information pertaining to the currently chosen routine appears on the screen. This information includes a brief description of the routine, a list of the required input data and very brief operating instructions for the routine. After the information has been read, the program returns the user to the menu from which HELP was selected.

If HELP is selected from DACSAP's main menu, a brief description of DACSAP appears on the screen followed by a menu of options. From this menu the use may choose to read any of the previously mentioned HELP files or print a hard-copy of the entire HELP routine. A sample hardcopy printout is contained in Appendix B.

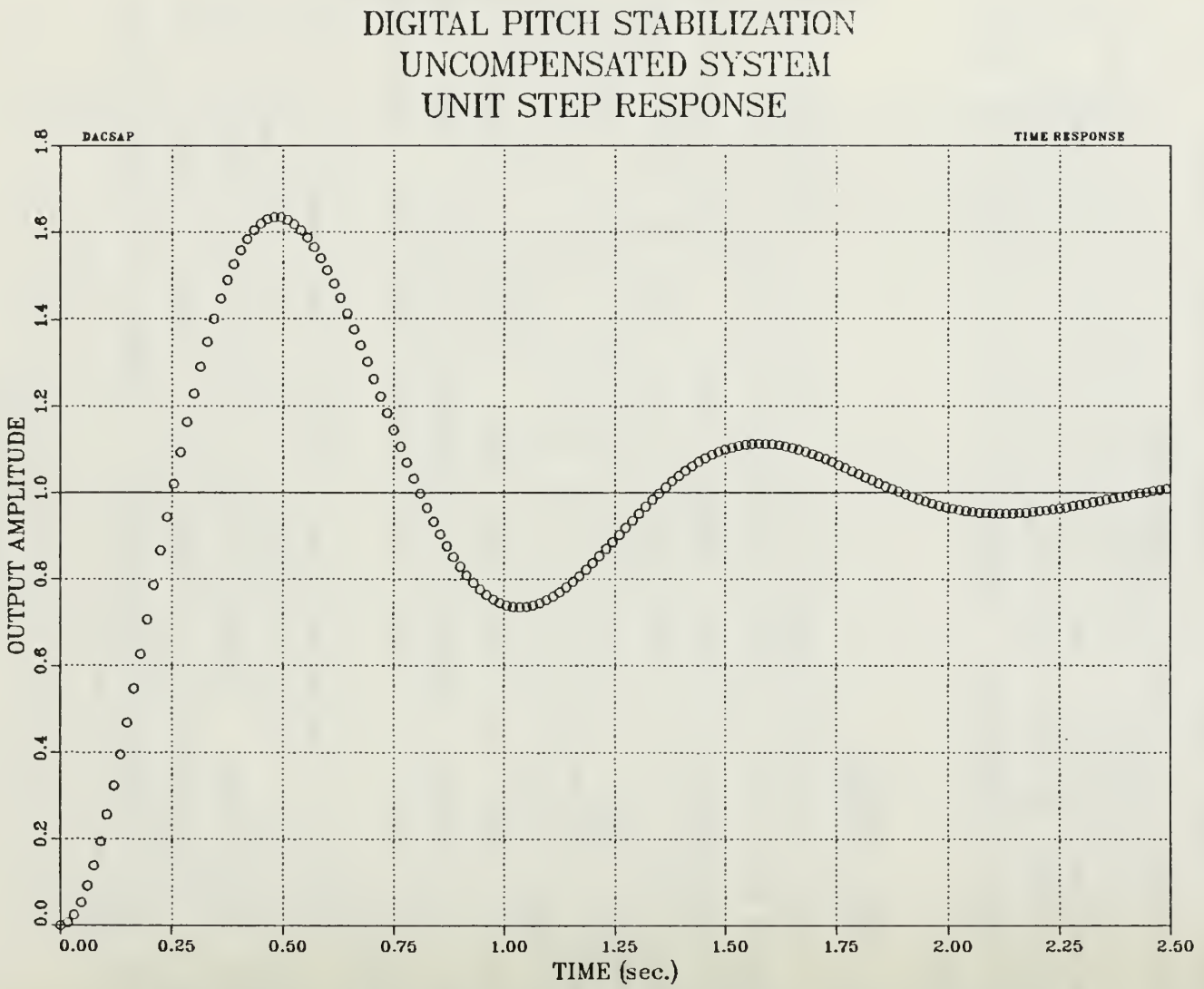


Figure 3.14 Discrete System Time Response Plot.

F. SUMMARY

In this chapter the capabilities of the major branches of DACSAP are covered as well as some of the concepts used in bringing about these capabilities. Appendix D contains a programmer's guide for DACSAP. An outline of each of DACSAP's subroutines is contained in this guide including definitions of the main variables and arrays and notes on the specific functions of each subroutine. Individuals who may wish to modify DACSAP in the future should find the programmer's guide and information in this chapter useful in analyzing the program.

IV. CONCLUSIONS AND RECOMMENDATIONS

The primary objective of this thesis was to develop a comprehensive computer aid for the design and analysis of linear control systems. The program was aimed at reducing or eliminating the large number of calculations involved in the design of control systems. To this end, the program was to allow for the entry of transfer functions as they appear in a block diagram, to incorporate all of the common control system analysis techniques and to permit the analysis of both continuous and digital control systems.

A. SUMMARY OF RESULTS

The result of the thesis is the Digital and Analog Control System Analysis Program or DACSAP. DACSAP is a fully interactive design aid which incorporates a wide variety of methods for block diagram manipulation and control system analysis. The following list outlines the program's capabilities:

1) Interactive, menu driven program control.

2) Block diagram manipulations including entry of transfer functions in coefficient or factored form, open and closed loop transfer function calculations and reduction of multi-locp systems.

3) Continuous and discrete root locus analysis including the interactive examination of individual root locations, tabular output and root locus plots.

4) Continuous and discrete frequency response analysis including bilinear frequency transformations, open and closed loop Bode plots, polar plots (Nyquist analysis), log magnitude-phase plots (Nichols analysis) and tabular output.

5) Continuous and discrete time response analysis including the response to step, impulse and ramp inputs, tabular output and time history plots.

6) Mass storage data file routines which allow the user to store the entire block diagram plus the open or closed loop transfer functions for use during a subsequent run of the program.

7) Graphical output is produced using a state-of-the-art graphics package (see Appendix C) which creates fast, high resolution plots at the user's terminal allowing real time analysis of the system and true interactive design.

8) A HELP routine which provides the user with a brief outline of operating instructions and available options for each of the program's major routines.

The program is currently available for use by all users of the Naval Postgraduate School's IBM 370 mainframe computer and is being used by students and faculty for course work and thesis research.

B. RECOMMENDATIONS

During the development of DACSAP, an attempt was made to cover a large number of the basic needs of the control system designer. There are many more capabilities that could be added to the program which would enhance its utility in specific areas. The following is a partial list of such capabilities:

1) The ability to input sampling period as a variable in discrete system transfer functions, thus allowing the designer to observe the effect of varying sampling rate without having to recalculate any transfer functions.

2) The capability to perform frequency response analysis of z-domain transfer functions. The routine must include the automatic transformation of z-domain transfer functions to the w-plane and the transformation required to present the response as a function of real (unwarped) frequency.

3) The ability to sum transfer functions and the ability to designate blocks in parallel feed forward paths.

4) Routines specifically oriented toward the design of gain, lead and lag comensators using root locus and frequency response techniques.

5) Conversion of discrete transfer functions to continuous and vice versa.

6) The ability to convert systems represented in state-space form to equivalent transfer functions.

Unfortunately, the average user at the Naval Postgraduate School is allotted only one megabyte of virtual

memory on the IBM 370 and DACSAP (in combination with the DISSPLA routines) requires nearly all of that memory to run. Therefore, any additions to the program would have to be at the expense of some existing routines or the structure of the program would have to be changed.

Since the list above involves mostly transfer function manipulations, one possibility for restructuring the program would be to separate the transfer function manipulation routines from the rest of the program (analysis and plotting routines). The results of the transfer function routines could be placed in a data file which could then be read by the analysis routines. However, this would reduce the flexibility and ease of operation of the program and increase its execution time.

Thus, room does exist for increasing the capabilities of DACSAP. However, some trade-offs will probably have to be made if additions are to be made to the program, and that program is to be available to all users at the Naval Postgraduate School.

APPENDIX A
USER'S MANUAL

Digital and Analog Control System Analysis Program
DACSAP

DACSAP is an interactive aid for the design and analysis of linear, single input / single output (SISO) control systems. The program is designed to manipulate transfer functions from a control system block diagram and create a variety of output plots or tabulated data suitable for analysis. Analysis techniques which may be used include root locus, Bode, Nyquist, Nichols and time response.

The program is fully interactive and easy to use, requiring the user to know no special language or commands. All data entry is prompted and options are presented to the user via a set of option menus. For these reasons, this manual is meant to be more of a tutorial for first time users. It is hoped that the user finds the program so easy to use that he no longer needs to refer to this manual after the program has been run a time or two. Towards this end, HELP routines may be accessed during program execution. The HELP routines briefly summarize the major points contained in this manual and may be used to refresh the experienced user's memory as to the operating procedures for DACSAP.

A. GETTING STARTED

At the U.S. Naval Postgraduate School, DACSAP resides as a DISSPLA text module on a separate disk which may be accessed by typing the word CONTROLS after logging on. Since DACSAP, in combination with the DISSPLA subroutines, takes nearly one megabyte of memory to operate, the user should define his virtual memory as one megabyte (1024 K) prior to accessing the CONTROLS disk. If this is not done, the CONTROLS disk MANAGE EXEC will define the proper storage, then halt execution. The user will then be required to reenter the CMS operating system and reaccess the CONTROLS disk. Once on the CONTROLS disk, DACSAP is executed by selecting it from the MANAGE EXEC menu.

When program execution has begun, the user will be asked to select an output device as shown below.

GRAPHICAL OUTPUT DEVICE OPTIONS:

1. TEK 618
2. TEK 4113 OR 4114
3. IBM 3278
4. DISSPLA METAFILE

PLEASE SELECT A GRAPHICAL OUTPUT DEVICE (1-4).

Since DACSAP is meant to be an interactive design tool, it is best utilized when output is to a high speed, high resolution device such as the TEK 618 or TEK 4113/4114 with a high speed modem. If one wishes to input a large system, make changes to a system or obtain tabular data prior to performing graphical analysis, and do so without tying up a graphics terminal, then the IBM 3278 terminal may be selected. Graphical output to these terminals is possible, but is in the form of an unlabelled, low resolution, line printer style plot. The general shape of curves can be observed, but detailed analysis of any kind is practically impossible.

Output to a DISSPLA metafile is available to those users who desire hardcopy, publication quality plots for use in theses, technical reports, etc. After the metafile has been formed the user is free to select any of a number of output devices available through the DISSPOP routine. The actual plots produced are of the proper size for thesis figures and would, therefore, fit comfortably in any document written on 8.5 X 11 inch paper. The figures for this document were produced using this option. The program may be run from any terminal if this option is chosen since no graphical output comes to the terminal.

B. TRANSFER FUNCTION INPUT

Before any analysis may be performed, it is first necessary to define the system. Each loop in the system may contain up to 20 blocks each of which may be of 20th order or less. However, the entire system must be of 100th order or less. In order to define the system, select Option 1 from the DACSAP Option menu below:

DACSAP OPTIONS	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	BODE ANALYSIS (OPEN LOOP)
4	BODE ANALYSIS (CLOSED LOOP)
5	NYQUIST ANALYSIS
6	NICHOLS ANALYSIS
7	TIME RESPONSE
8	CHANGE BLOCK DIAGRAM
9	SAVE THIS PROBLEM
10	START A NEW PROBLEM
11	HELP
12	EXIT DACSAP

Selecting Option 1 will produce the following menu of transfer function input options:

TRANSFER FUNCTION INPUT OPTIONS	
OPTION NO.	OPTION
1	ENTER XFER FUNCTION(S) FROM CONSOLE
2	LOAD TRANSFER FUNCTION(S) FROM A FILE
3	RETURN TO DACSAP MENU
4	HELP

Option 2 from the menu should be selected if the user has saved a system during a previous terminal session and would now like to continue work on that system. The user will be prompted for the filename, then the system will be loaded and transfer function input will be complete.

Option 1 should be selected if the system is being defined for the first time and will result in the following questions:

IN WHICH DOMAIN ARE YOU WORKING? (S,Z,W, OR WP)

WHICH LOOP ARE YOU PREPARING TO INPUT?

HOW MANY BLOCKS ARE IN YOUR OPEN-LOOP?

It should be noted that blocks which consist of purely a gain should not be entered as transfer functions. The program requests values for open loop gain, forward path gain and feedback path gain whenever they are needed for calculations. The following questions are asked for each of the blocks in the open loop:

TRANSFER FUNCTION NO. 1

IS THIS BLOCK IN THE FORWARD OR FEEDBACK PATH? (F OR B)

DO YOU WISH TO INPUT THE TRANSFER FUNCTION FOR THIS BLOCK FROM A DATA FILE OR FROM THE CONSOLE? (D OR C)

If entering the block's transfer function from the console, the following questions will be asked:

WHAT IS THE ORDER OF THE NUMERATOR POLYNOMIAL?

WHAT IS THE ORDER OF THE DENOMINATOR?

IS THIS TRANSFER FUNCTION IN FACTORED OR COEFFICIENT FORM? (F OR C)

1. Transfer Functions in Factored Form

The numerator and denominator of each block must be of the same form. For transfer functions in factored form, input is prompted as follows:

WHAT IS THE NUMERATOR GAIN CONSTANT?

WHAT ARE THE ROOTS OF THE NUMERATOR?

ROOT NO. 1
REAL PART =

IMAGINARY PART =

The last three lines are repeated a number of times equal to the order of the polynomial. After the numerator roots are

entered, they are displayed as follows allowing the user to check for and correct any typing errors.

SUMMARY OF ROOTS		
LOOP NUMBER 1 BLOCK NUMBER 1		
ITEM NO.	NUMERATOR ROOTS	
1	-0.0300000	0.0000000 J
2	-4.3000000	2.5780000 J
3	-4.3000000	-2.5780000 J
4	CONSTANT =	25.0000000

ANY CHANGES TO THESE PARAMETERS?

Entry of the denominator polynomial is identical to that of the numerator.

2. Transfer Functions in Coefficient Form

If the transfer function is in coefficient form, the user will be prompted as follows:

```
NUMERATOR COEFFICIENTS.
     COEFFICIENT CF S  **  2 =
     COEFFICIENT OF S  **  1 =
     COEFFICIENT CF S  **  0 =
```

Naturally, the number of coefficients requested will correspond to the order of the polynomial (the case above would be of second order). After they have been input, the coefficients are displayed as follows and may be corrected if necessary.

SUMMARY OF CCEFFICIENTS		
LOOP NUMEER 1 BLOCK NUMBER 2		
ITEM NO.	NUMERATOR COEFFICIENTS	
1	1.0000000	S ** 2
2	-0.3600000	S ** 1
3	0.1600000	S ** 0

ANY CHANGES TO THESE PARAMETERS?

Entry of the denominator polynomial is identical to that of the numerator.

3. Transfer Functions in a Data File

If an inner feedback loop, a compensator, a filter or any other subsystem had been designed and saved using DACSAP in a previous terminal session, its transfer function could now be loaded into the appropriate block of the current system. This allows the user to design subsystems independently of the overall system then install them later. The user is simply prompted for the filename under which the subsystem is saved.

C. ROOT LOCUS ANALYSIS

Root locus plots or a tabulation of root locations may be obtained by selecting Option 2 from the DACSAP option menu shown below.

DACSAP OPTIONS	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
> 2	ROOT LOCUS ANALYSIS <
3	BODE ANALYSIS (OPEN LOOP)
4	BODE ANALYSIS (CLOSED LOOP)
5	NYQUIST ANALYSIS
6	NICHOLS ANALYSIS
7	TIME RESPONSE
8	CHANGE BLOCK DIAGRAM
9	SAVE THIS PROBLEM
10	START A NEW PROBLEM
11	HELP
12	EXIT DACSAP

This selection will yield the following menu of root locus options:

ROOT-LOCUS OPTIONS	
OPTION NO.	OPTION
1	ROOT-LOCUS PLOT
2	TABULATED DATA
3	RETURN TO DACSAP MENU
4	EXIT DACSAP
5	HELP

If a plot is desired, Option 1 should be selected. The user is then prompted for information in three areas as follows:

```
GAIN PARAMETERS
     MIN GAIN VALUE =
     MAX GAIN VALUE =
     POSITIVE OR NEGATIVE FEEDBACK? (P OR N)
     TIC MARKS? (Y OR N)
     WHAT IS YOUR SAMPLE PERIOD?
         T =          (discrete systems only)
```

```
PLOTTING DIMENSIONS
     X MAX =
     X MIN =
     Y MAX =
     Y MIN =

PLOT HEADING
     HOW MANY LINES OF HEADING WOULD YOU LIKE? (4 MAX)
     A MAXIMUM OF 32 CHARACTERS PER LINE IS ALLOWED.
     LINE 1 =    (repeated as required)
```

If the user is unsure of how to dimension the root locus plot, it may help to first select Option 2 of the Root Locus Options menu. This will produce the following information at the terminal: (sample values)

```
-------------------------------------------------------
|      OPEN-LOOP TRANSFER FUNCTION COEFFICIENTS       |
-------------------------------------------------------
| DEGREE |       NUMERATOR       |     DENOMINATOR    |
-------------------------------------------------------
|    4   |                       |          1.0000000 |
|    3   |                       |          9.0400000 |
|    2   |          25.0000000   |          0.3760009 |
|    1   |          25.7499847   |          0.2079997 |
|    0   |           0.7499999   |          0.5759997 |
-------------------------------------------------------

POLE-ZERO LOCATIONS
     POLES:
          REAL PART          IMAG. PART
       -0.89999924E+01    0.00000000E+00
       -0.40000027E+00    0.00000000E+00
        0.17999995E+00    0.35721129E+00
        0.17999995E+00   -0.35721129E+00

     ALL ZEROES ARE AT INFINITY EXCEPT FOR THE FOLLOWING:
          REAL PART          IMAG. PART
       -0.99999934E+00    0.00000000E+00
       -0.30000009E-01    0.00000000E+00

WOULD YOU LIKE A TABULAR OUTPUT OF ROOT LOCATIONS?  (Y OR N)
```

A detailed listing of root locations may be obtained by answering yes (Y) to the last question. By answering no (N) to the question, the user is returned to the Root Locus Options menu and may choose the option to produce a plot

(Option 1). The tabulation of pole-zero locations should help in choosing the dimensions for the plot. It should be noted that the plot area is always square. So, the dimensions should also be chosen to be square (i.e., XMAX-XMIN=YMAX-YMIN) if true angular representation is desired.

When the root locus plot has been completed, the following question will appear on the screen:

```
WOULD YOU LIKE INFORMATION ABOUT A PARTICULAR POINT
ON THE ROOT-LOCUS? (Y OR N)
```

By replying yes (Y), information concerning any root location may be presented by responding to the following prompts:

```
WHICH ROOT LOCATION?
     REAL PART =

     IMAGINARY PART =
```

The following information will be provided for that root location: (sample values)

```
GAIN =        1.5746107
NATURAL FREQUENCY =        5.3740110  RAD/SEC
DAMPING RATIO = 0.7071068

ANOTHER POINT? (Y OR N)
```

More root locations may be examined by responding with a yes (Y) to the last question. When a negative response is input, the following question is asked:

```
WOULD YOU LIKE TO HIGHLIGHT A PARTICULAR GAIN VALUE?
```

If a yes (Y) is input, the user is asked for a gain value and a small box is drawn around the root locations associated with that gain value. At this point the root locus plot is complete. Pole-zero locations and the stable region crossover gain are then presented at the terminal. The

stable region crossover gain is that gain at which the root locus first crosses from the stable region to the unstable region.

Lines of constant damping ratio and natural frequency are drawn on plots for discrete systems. For w- and w'-domain plots only a few lines are drawn, just to make the user aware of the behavior of these characteristics. Due to the complexity of these lines on the z-plane, they are drawn and labelled in greater detail. This should help make these plots easier analyze. Of course, the most detailed information concerning damping ratio and natural frequency may be obtained after the plot is complete by examining individual points, as described in the previous paragraph. Figures A.1 and A.2 are examples of DACSAP root locus plots.

It should be noted that if one or more lines of heading are specified, the plot will be rotated on its side so that the long edge of the plot is aligned with the long edge of the screen. With the plot oriented in such a way, a hardcopy print of the screen will better fill the page. If the user intends to run a multitude of plots, conducting analysis on the screen, it is suggested that the plots be produced without headings until a hardcopy is desired. This will reduce the plotting time and produce a larger, upright plot to analyze on the screen.

Subsequent selection of Option 1 from the Root Locus Options menu will result in the following menu of change options:

ROOT-LOCUS CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE TRANSFER FUNCTION(S)
2	CHANGE GAIN PARAMETERS
3	CHANGE PLOTTING PARAMETERS
4	CHANGE PLOT HEADING
5	NO CHANGES (READY TO PLOT)
6	RETURN TO DACSAP MENU

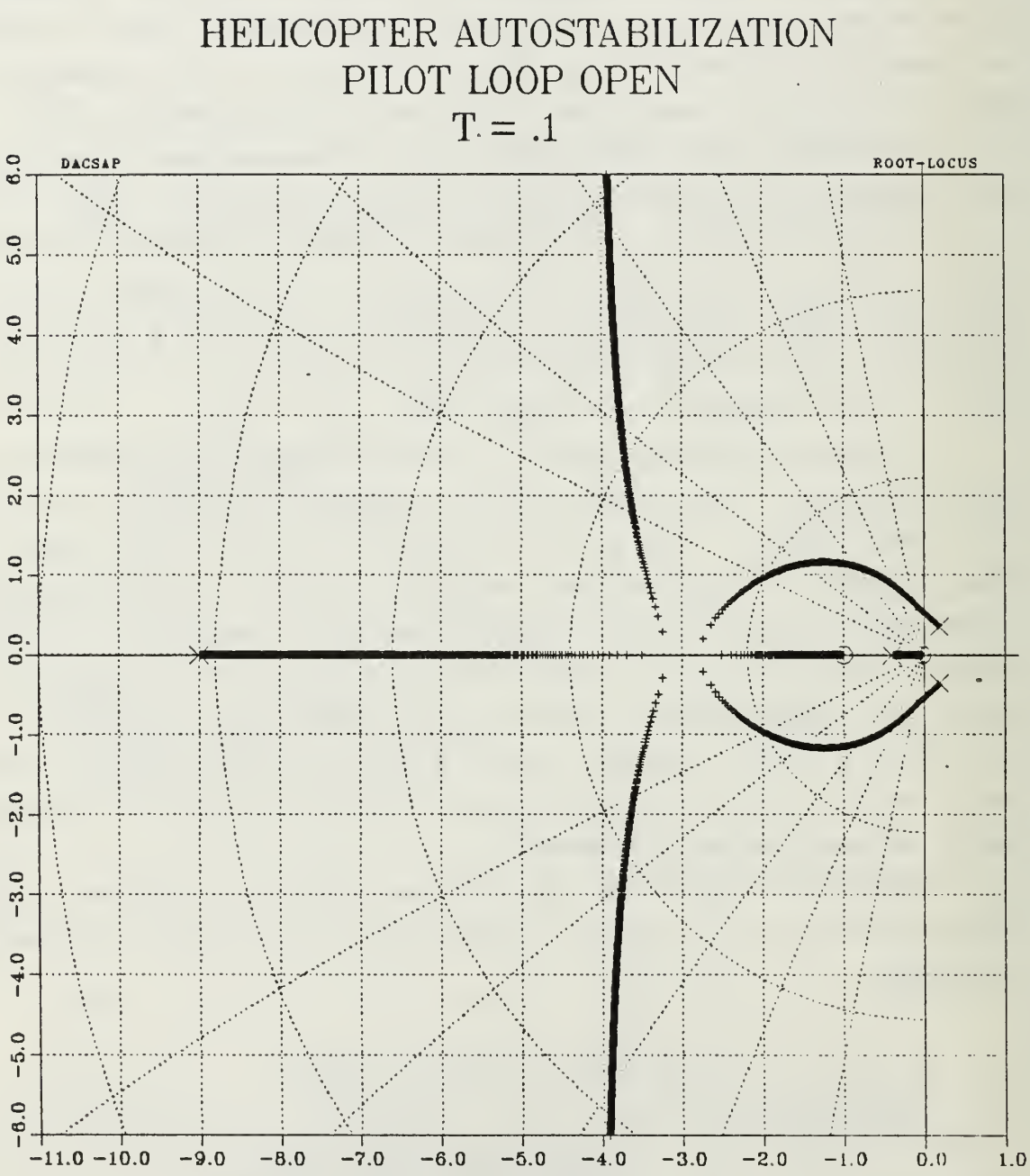


Figure A.1 Sample w'-plane Root Locus Plot.

DIGITAL ANTENNA AZIMUTH CONTROL

COMPENSATED SYSTEM

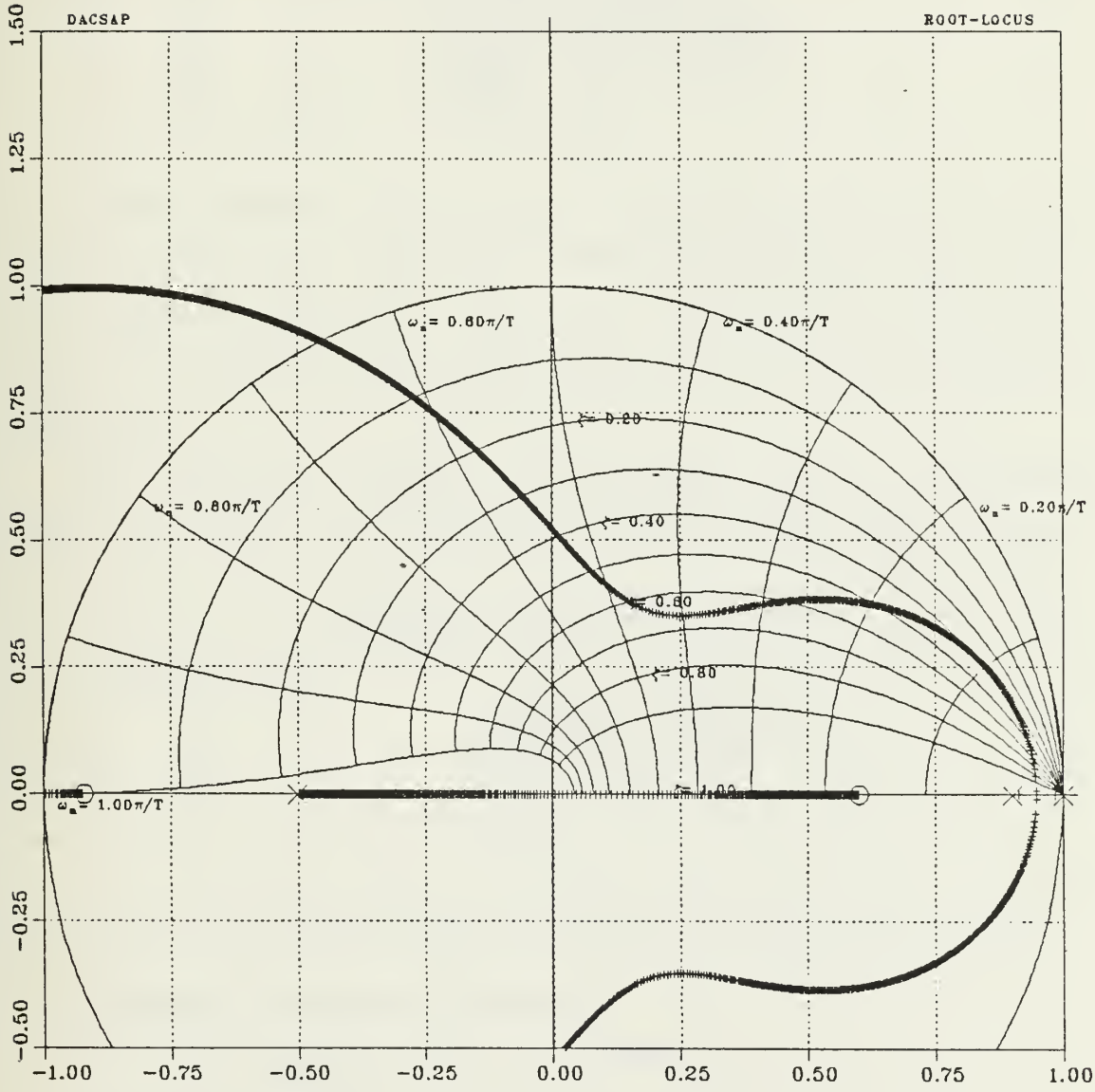


Figure A.2 Sample z-plane Root Locus Plot.

From this menu the user may selectively change any or all of the parameters required to produce another root locus plot.

D. FREQUENCY RESPONSE ANALYSIS

	DACSAP OPTIONS		
	OPTION NO.	OPTION	
	1	INPUT TRANSFER FUNCTION(S)	
	2	ROOT LOCUS ANALYSIS	
>	3	BODE ANALYSIS (OPEN LOOP)	<
>	4	BODE ANALYSIS (CLOSED LOOP)	<
>	5	NYQUIST ANALYSIS	<
>	6	NICHOLS ANALYSIS	<
	7	TIME RESPONSE	
	8	CHANGE BLOCK DIAGRAM	
	9	SAVE THIS PROBLEM	
	10	START A NEW PROBLEM	
	11	HELP	
	12	EXIT DACSAP	

DACSAP offers four different frequency domain analysis tools. They are open and closed loop Bode plots, polar plots (for Nyquist analysis) and log magnitude-phase plots (for Nichols analysis). Options 3 through 6 of the DACSAP Options menu should be used to select the desired analysis technique.

The Frequency Response Options menu, below, is presented when any of the four frequency domain methods are chosen.

FREQUENCY RESPONSE OPTIONS	
OPTION NO.	OPTION
1	GRAPHICAL ANALYSIS
2	TABULATED DATA
3	RETURN TO DACSAP MENU
4	EXIT DACSAP
5	HELP

For the three open loop analysis methods (Bode, Nyquist and Nichols) the following information will be requested:

```
FREQUENCY RESPONSE PARAMETERS:
     WHAT IS YOUR OPEN-LOOP GAIN?
          GAIN =
FREQUENCY RANGE:
     MIN FREQUENCY =    (RAD/SEC)
     MAX FREQUENCY =    (RAD/SEC)
```

PLOT HEADING

HOW MANY LINES OF HEADING WOULD YOU LIKE? (4 MAX)

A MAXIMUM OF 32 CHARACTERS PER LINE IS ALLOWED.
LINE 1 = (repeated as required)

The prompts for closed loop Bode analysis are the same except that instead of OPEN LOOP GAIN the user is asked for the following:

WHAT IS THE FORWARD PATH GAIN?

WHAT IS THE FEEDBACK PATH GAIN?

POSITIVE OR NEGATIVE FEEDBACK? (P OR N)

The more detailed information requested for closed loop Bode plots is required so that the closed loop transfer function may be calculated.

If a discrete time system is being analyzed, the following question will also be asked:

FICTITIOUS (V) OR REAL (W) FREQUENCIES? (V OR W)

Since w- and w'-transformations result in frequency "warping", a frequency transformation must take place if the response is to be plotted as a function of real frequency. DACSAP will perform the transformation if W (omega) is selected or plot the response as a function of fictitious frequency if V (nu) is chosen. If W is selected the user will be prompted for the sample period which is required to perform the frequency transformation.

Once all of the parameters have been input, the frequency response will be plotted using the type of plot selected from the DACSAP Option menu. Figures A.3, A.4 and A.5 are examples of DACSAP frequency response plots. Frequency response plots are automatically scaled, so the user is not prompted for plot dimensions.

When these plots are complete, information concerning phase and gain margins and crossover frequencies is presented as follows: (sample values)

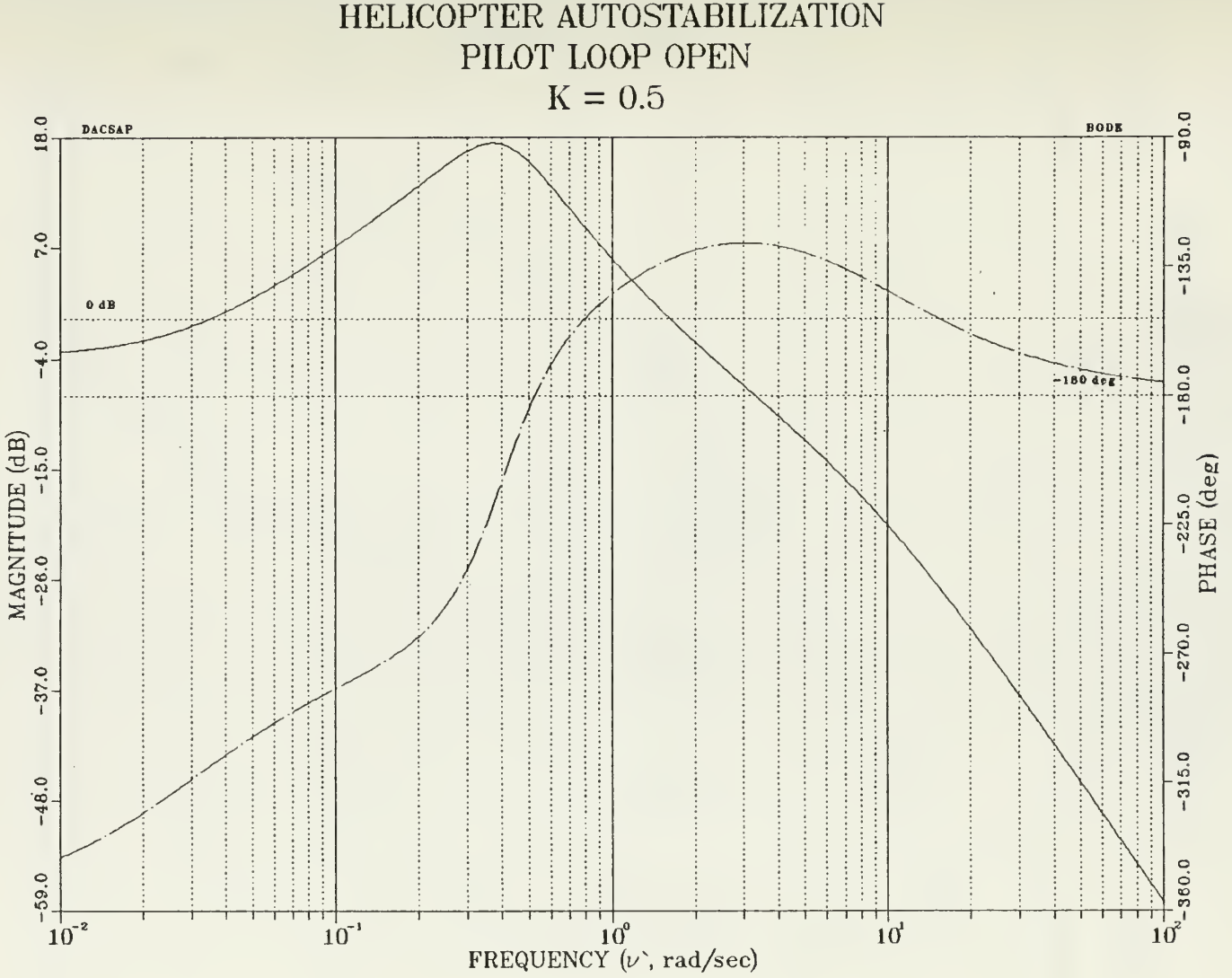


Figure A.3 Sample Bode Plot.

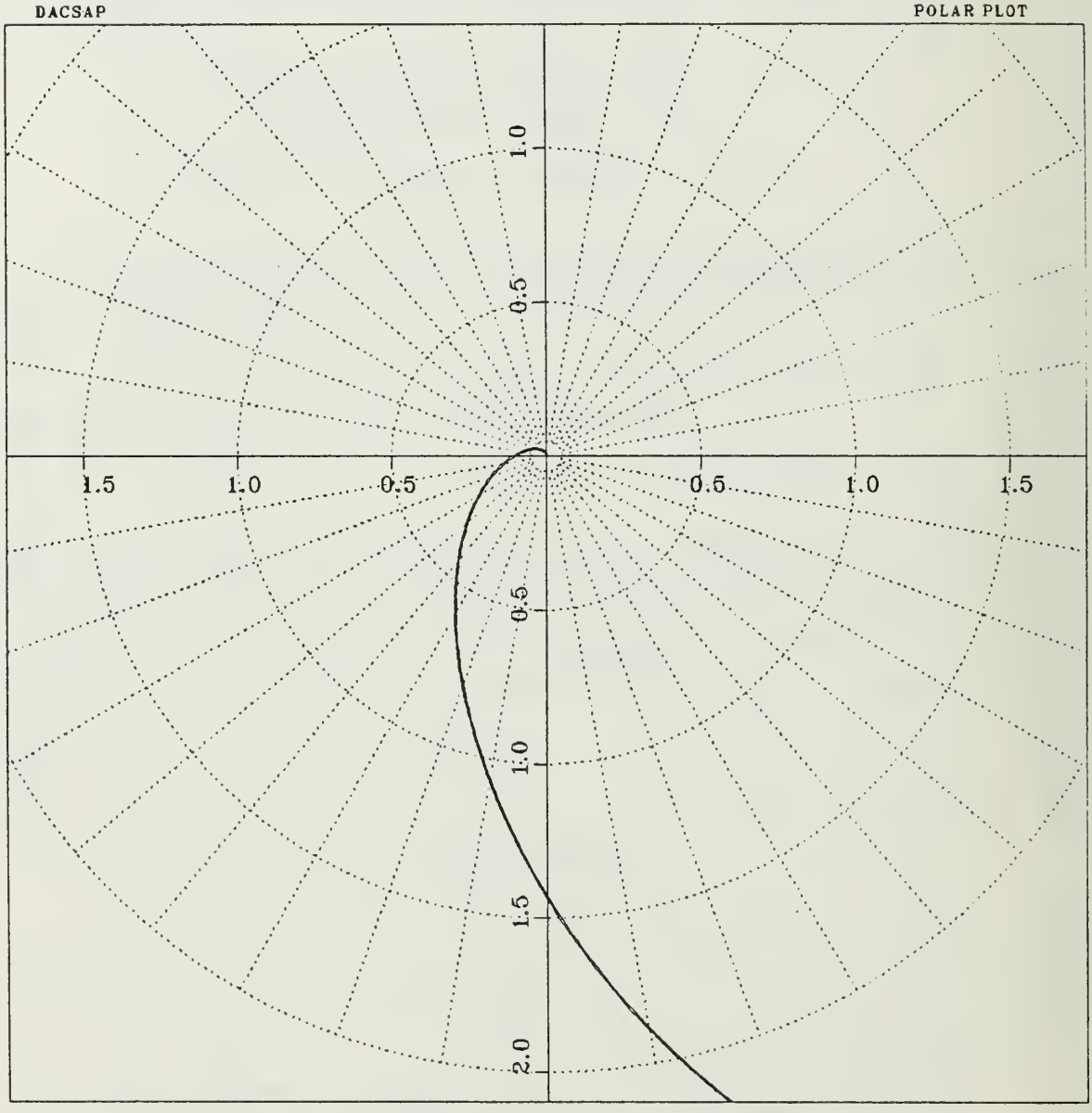


Figure A.4 Sample Polar Plot.

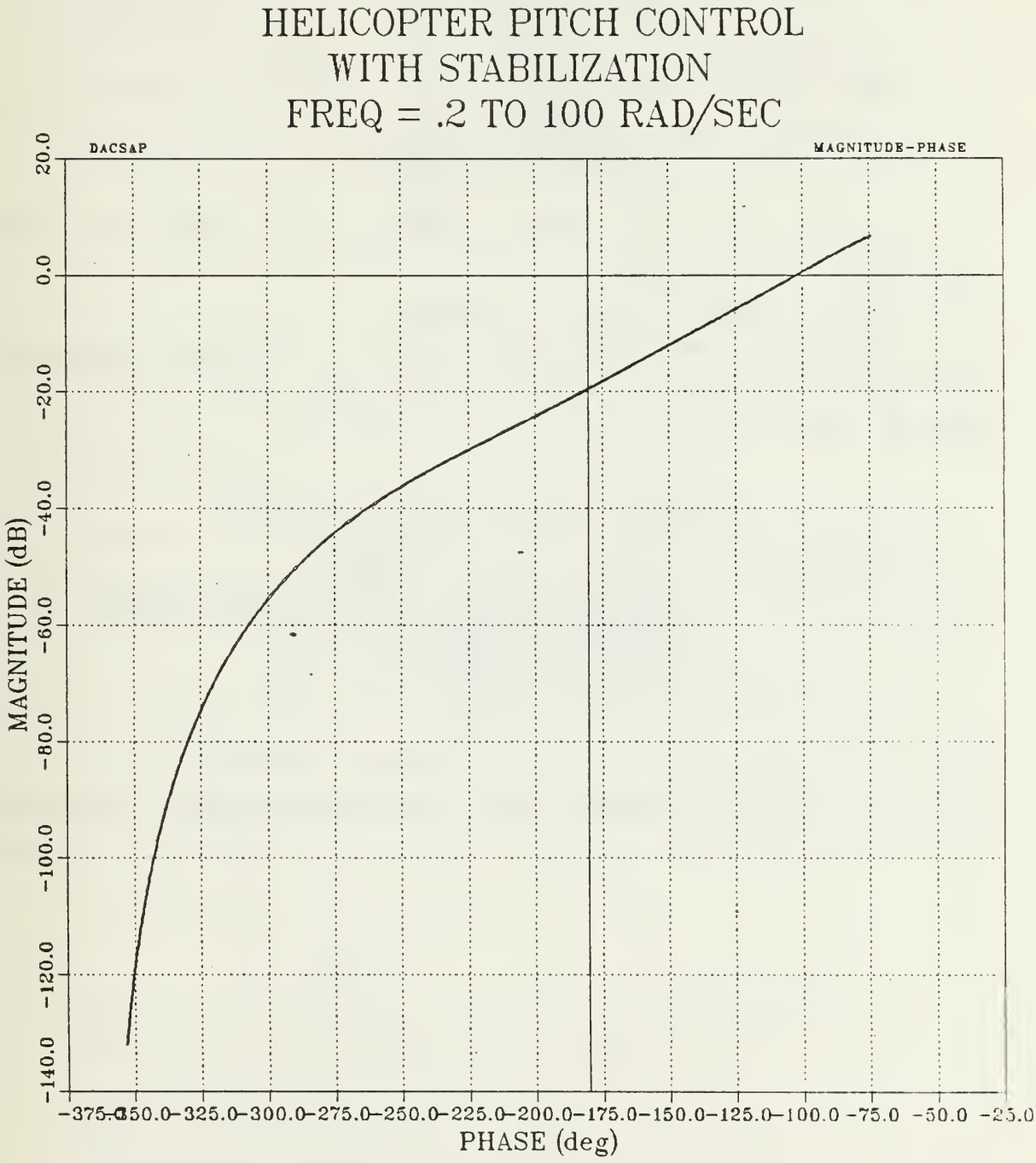


Figure A.5 Sample Log Magnitude-Phase Plot.

OPEN-LOOP TRANSFER FUNCTION COEFFICIENTS		
DEGREE	NUMERATOR	DENOMINATOR
4		1.0000000
3		9.0400000
2	25.0000000	0.3760009
1	25.7499847	0.2079997
0	0.7499999	0.5759997

PHASE CROSSOVER FREQUENCY = 0.5299916E+00 RAD/SEC
GAIN MARGIN = -24.85826 DB

GAIN CROSSOVER FREQUENCY = 0.4135011E+01 RAD/SEC
PHASE MARGIN = 51.81558 DEGREES

As with root locus plots, polar and log magnitude-phase plots are rotated on their sides if one or more lines of heading are specified for the plot.

Subsequent selection of Option 1 from the Frequency Response Options menu will result in the following menu of change options:

FREQUENCY RESPONSE CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE TRANSFER FUNCTION(S)
2	CHANGE FREQUENCY RESPONSE PARAMETERS
3	CHANGE PLOT HEADING
4	NO CHANGES (READY TO PLOT)
5	RETURN TO DACSAP MENU

From this menu the user may selectively change any or all of the parameters required to produce another frequency response plot.

E. TIME RESPONSE ANALYSIS

DACSAP OPTIONS	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	BODE ANALYSIS (OPEN LOOP)
4	BODE ANALYSIS (CLOSED LOOP)
5	NYQUIST ANALYSIS
6	NICHOLS ANALYSIS
> 7	TIME RESPONSE <
8	CHANGE BLOCK DIAGRAM
9	SAVE THIS PROBLEM
10	START A NEW PROBLEM
11	HELP
12	EXIT DACSAP

A time history of the total system's response to a user specified input may be obtained by selecting Option 7 from the DACSAP Option menu shown above. This selection will result in the presentation of the following menu of time response options:

TIME RESPONSE OPTIONS	
OPTION NO.	OPTION
1	GRAPHICAL ANALYSIS
2	TABULATED DATA
3	RETURN TO DACSAP MENU
4	EXIT DACSAP
5	HELP

Selecting Options 1 or 2 will result in the user being prompted for the following parameters:

```
TIME RESPONSE PARAMETERS:

          1 = STEP
          2 = IMPULSE
          3 = RAMP
SELECT TYPE OF INPUT TO YOUR SYSTEM (1,2 OR 3)

WHAT IS THE AMPLITUDE OF YOUR INPUT?

WHAT IS THE FORWARD PATH GAIN?

WHAT IS THE FEEDBACK PATH GAIN?

POSITIVE OR NEGATIVE FEEDBACK?  (P OR N)

WHAT IS YOUR SAMPLE PERIOD?  T = (SEC)
(discrete systems)
```

FOR HOW MANY SECONDS DO YOU WISH TO SEE THE RESPONSE?

If Option 1 was selected from the Time Response Options menu, the user will also be prompted for a plot heading. Time response plots are automatically scaled, so the user is not prompted for plot dimensions. If Option 2 was chosen, the output may be selected to go to the screen or the printer. Figures A.6 and A.7 are examples of DACSAP time response plots.

Subsequent selection of Option 1 from the Time Response Options menu will result in the following menu of change options:

TIME RESPONSE CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE TRANSFER FUNCTION(S)
2	CHANGE TIME RESPONSE PARAMETERS
3	CHANGE PLOT HEADING
4	NO CHANGES (READY TO PLOT)
5	RETURN TO DACSAP MENU

From this menu the user may selectively change any or all of the parameters required to produce another frequency response plot.

TACTICAL AIRCRAFT YAW DAMPER

K = .3218

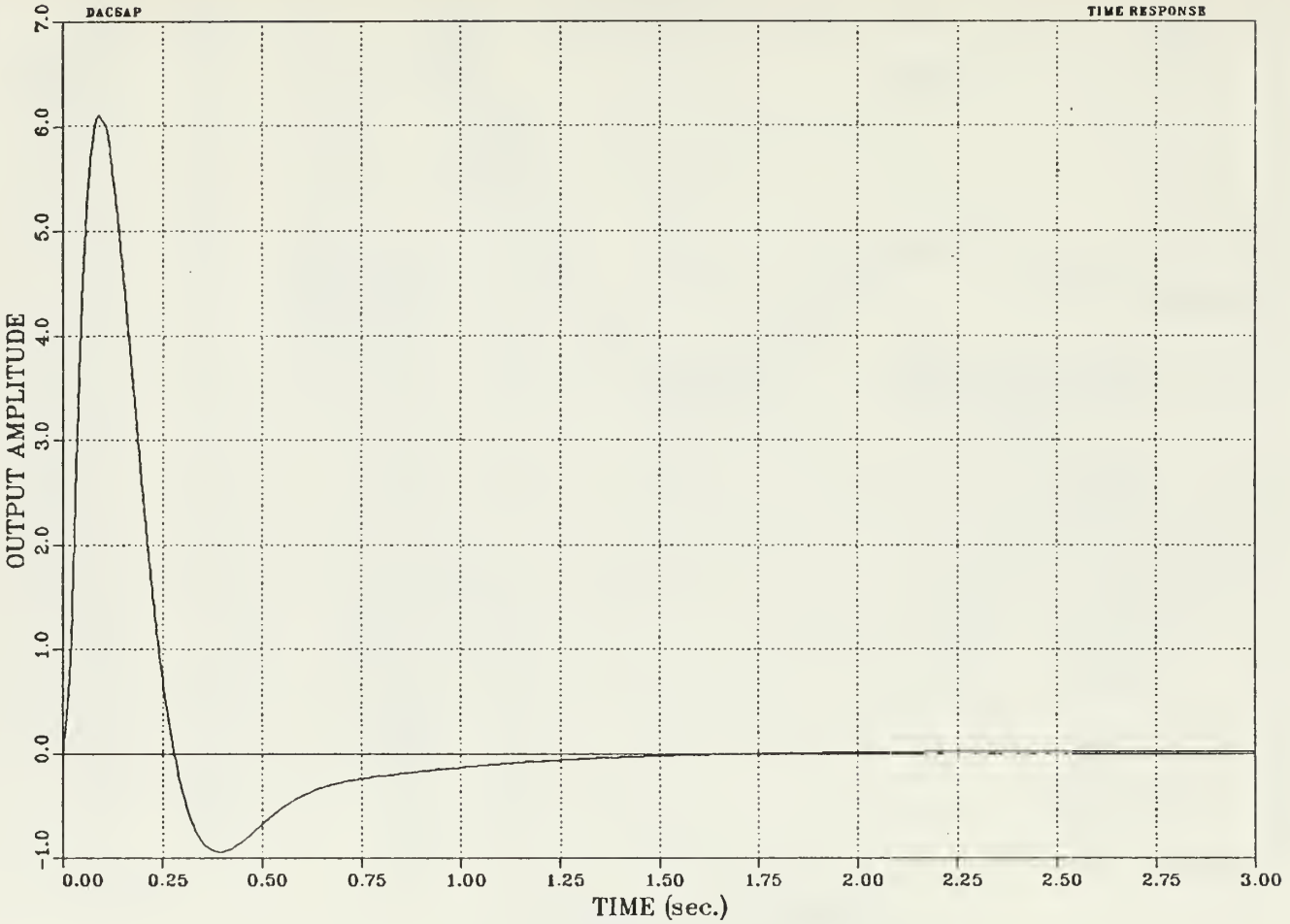


Figure A.6 Continuous System Time Response Plot.

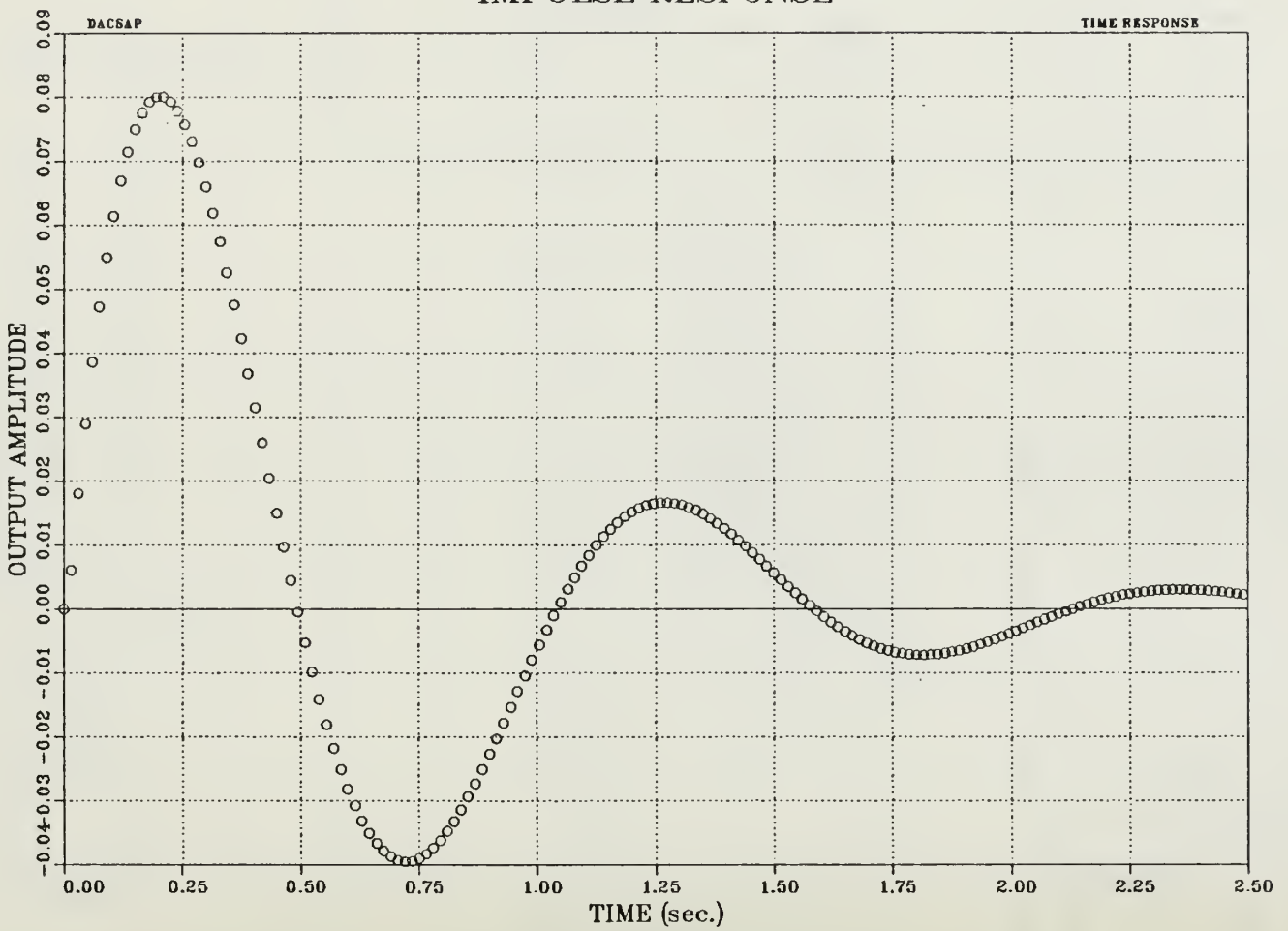


Figure A.7 Discrete System Time Response Plot.

F. CHANGING THE BLOCK DIAGRAM

In order to change or add to the transfer functions in the block diagram, Option 8 of DACSAP's option menu should be selected.

DACSAP OPTIONS	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	BODE ANALYSIS (OPEN LOOP)
4	BODE ANALYSIS (CLOSED LOOP)
5	NYQUIST ANALYSIS
6	NICHOLS ANALYSIS
7	TIME RESPONSE
> 8	CHANGE BLOCK DIAGRAM <
9	SAVE THIS PROBLEM
10	START A NEW PROBLEM
11	HELP
12	EXIT DACSAP

Entry to this change routine can also be obtained by selecting Option 1 of the root locus, frequency response or time response parameter change menus.

Transfer function change options are presented in the following menu:

TRANSFER FUNCTION (T.F.) CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE A T.F. IN THE CURRENT LOOP
2	ADD A NEW BLOCK TO THE CURRENT LOOP
3	CHANGE T.F. IN AN INNER LOOP
4	EXPAND TO AN OUTER LOOP
5	NO CHANGES
6	HELP

Each of these options will be discussed separately.

1. Changing Transfer Functions in the Current Loop

When Option 1 of the Transfer Function Change menu is selected, the user will be asked which block is to be

changed. Then the following menu of block change options will be presented:

```
BLOCK CHANGE OPTIONS
BLOCK NUMBER  1

| OPTION NO. |                    OPTION                    |
|------------|----------------------------------------------|
|     1      | CHANGE CURRENT NUMERATOR                     |
|     2      | CREATE A NEW NUMERATOR                       |
|     3      | CHANGE CURRENT DENOMINATOR                   |
|     4      | CREATE A NEW DENOMINATOR                     |
|     5      | CREATE NEW NUMERATOR & DENOMINATOR           |
|     6      | MOVE BLOCK TO THE FEEDBACK PATH              |
|     7      | NO MORE CHANGES TO THIS BLOCK                |
```

If Options 1 or 3 are selected, a summary of the roots or coefficients of the selected polynomial will be presented in one of the forms shown below:

```
SUMMARY OF ROOTS
LOOP NUMBER  1
BLOCK NUMBER  1

| ITEM NO. |          NUMERATOR ROOTS          |
|----------|-----------------------------------|
|     1    |  -0.0300000        0.0000000 J    |
|     2    | CONSTANT =        25.0000000       |
ANY CHANGES TO THESE PARAMETERS?

SUMMARY OF COEFFICIENTS
LOOP NUMBER  3
BLOCK NUMBER  2

| ITEM NO. |        NUMERATOR COEFFICIENTS       |
|----------|-------------------------------------|
|     1    |   1.0000000  S **  2                |
|     2    |  -0.3600000  S **  1                |
|     3    |   0.1600000  S **  0                |
ANY CHANGES TO THESE PARAMETERS?
```

By answering yes (Y) to the question following the summary, the user may change any of the parameters displayed. A negative response will return the user to the Block Change Options menu. This method may be used if the user simply wishes to review the contents of a block.

If Options 2 or 4 are selected from the Block Change Options menu, the user will be prompted for all new parameters for the selected polynomial. The form of the polynomial (factored or coefficient) will remain the same as it was before. However, all other parameters may be changed including the order of the polynomial. The user will be prompted for the new parameters as he was in the transfer function input routine.

If the user wishes to change the entire character of a block, Option 5 of the Block Change menu should be selected. This option essentially erases the current parameters associated with the selected block and prompts the user for all new parameters. The user is free to input the transfer function in any form and entry may be from the console or from a data file. The prompts are exactly as described in the Transfer Function Input section of this manual.

Option 6 of the Block Change Options menu simply allows the user to redesignate a block as being in the feedback path if it is currently in the forward path or vice versa. Option 6 will read

| 6 | MOVE BLOCK TO THE FORWARD PATH |

for blocks which are currently in the feedback path.

When no more changes to the currently selected block are required, Option 7 should be selected. The user is then returned to the Transfer Function Change menu.

2. Adding a Block to the Current Loop

TRANSFER FUNCTION (T.F.) CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE A T.F. IN THE CURRENT LOOP
> 2	ADD A NEW BLOCK TO THE CURRENT LOOP <
3	CHANGE T.F. IN AN INNER LOOP
4	EXPAND TO AN OUTER LOOP
5	NO CHANGES
6	HELP

By selecting Option 2 from the Transfer Function Change menu, above, the user is simply sent to the transfer function input routine to enter the parameters for the new block. The prompts are exactly as described in the Transfer Function Input section of this manual. The transfer function may be of any form, it may be placed in the forward or feedback path and may be entered from the console or from a data file.

3. Changing Transfer Functions in an Inner Loop

If the user is currently working on an outer loop of a multiloop system where the inner loop(s) were designed and saved using DACSAP, then the inner loop(s) may be returned to and changed. Later it will be shown how to have these changes reflected in the outer loop(s).

TRANSFER FUNCTION (T.F.) CHANGE OPTIONS	
OPTION NO.	OPTION
1	CHANGE A T.F. IN THE CURRENT LOOP
2	ADD A NEW BLOCK TO THE CURRENT LOOP
> 3	CHANGE T.F. IN AN INNER LOOP <
4	EXPAND TO AN OUTER LOOP
5	NO CHANGES
6	HELP

When Option 3 of the Transfer Function Change Options menu is selected the user is prompted as follows:

```
IF YOU ARE GOING TO WANT TO COME BACK TO THIS LOOP
LATER, YOU SHOULD SAVE IT.

DO YOU WISH TO SAVE THE TRANSFER FUNCTION(S) IN
THIS LOOP? (Y OR N)
```

As stated, the current loop must be saved if it is to be returned to after changes are made to the inner loop. If it is not saved, its transfer functions will have to be re-entered from the console. The following subsection will cover how to get back to the outer loop(s).

When the user selects to save the current loop he will be prompted for a filename under which it is to be saved. After the outer loop is saved, the user will be prompted for the filename of the inner loop, that loop will be loaded and the following question will appear at the terminal:

```
WANT TO MAKE ANY CHANGES TO YOUR TRANSFER FUNCTION(S)?
     (Y OR N)
```

If a negative response is input, DACSAP's main option menu will be presented and an analysis technique for the inner loop may be selected. An affirmative response will result in the Transfer Function Change menu being presented and the user is free to change or add to the inner loop's transfer functions. It should be noted that if nothing but the forward or feedback path gains are to be changed, Option 4 of the Transfer Function Change menu should be selected to re-expand to the outer loop. The opportunity to change these gains will be given at that time, as explained in the following subsection.

4. Expanding to an Outer Loop

If the system currently being worked on is an inner loop of a multi-loop system, the user may expand the system to an outer loop, letting DACSAP calculate the inner loop

closed loop transfer function and place that transfer function anywhere in the outer loop. This operation can be made to occur by selecting Option 4 from the Transfer Function Change menu.

```
-------------------------------------------------------
|      TRANSFER FUNCTION (T.F.) CHANGE OPTIONS        |
-------------------------------------------------------
| OPTION NO.|                  OPTION                 |
-------------------------------------------------------
|     1     | CHANGE A T.F. IN THE CURRENT LOOP       |
|     2     | ADD A NEW BLOCK TO THE CURRENT LOOP     |
|     3     | CHANGE T.F. IN AN INNER LOOP            |
>    4     | EXPAND TO AN OUTER LOOP                 | <
|     5     | NO CHANGES                              |
|     6     | HELP                                    |
-------------------------------------------------------
```

Before expansion to the outer loop occurs, the user is given the opportunity to save the inner loop to a data file. This should be done if the inner loop is ever to be returned to for further analysis or changes. Prompting will be as follows:

```
IF YOU THINK YOU MIGHT WANT TO GO BACK AND MAKE
CHANGES TO THE CURRENT LOOP, YOU SHOULD SAVE IT.

DO YOU WISH TO SAVE THE TRANSFER FUNCTION(S) IN
THIS LOOP? (Y OR N)

WHAT NAME DO YOU WISH TO GIVE TO YOUR DATA FILE?
(8 CHARACTERS MAX)
```

```
-------------------------------------------------------
|              CLOSED LOOP PARAMETERS                 |
-------------------------------------------------------
| ITEM NO. |          PARAMETER          |   VALUE    |
-------------------------------------------------------
|     1    | FORWARD PATH GAIN           |    1.000   |
|     2    | FEEDBACK PATH GAIN          |    1.575   |
|     3    | POS. OR NEG. FEEDBACK       |      N     |
-------------------------------------------------------
THE CLOSED LOOP TRANSFER FUNCTION WILL ALSO BE
CALCULATED AND SAVED.

ANY CHANGES TO THESE PARAMETERS?
```

In the case shown above, the closed loop parameters had already been defined during the analysis of the inner loop. So, the user is simply given a last opportunity to change them. If they had not already been defined, the user would be prompted for them at this time.

At this point it is time to define the outer loop. The following prompt will begin the process:

```
DO YOU WISH TO INPUT THE OUTER LOOP TRANSFER FUNCTION(S)
FROM A DATA FILE OR FROM THE CONSOLE?  (D OR C)
```

Entry from the console will be discussed first.

a. Entering the Outer Loop From the Console

The first time that the system is expanded, the outer loop must be entered from the console. The user will be prompted as follows:

```
INTO WHICH BLOCK OF THE NEW LOOP DO YOU WISH TO PLACE
THE CLOSED LOOP TRANSFER FUNCTION OF THE INNER LOOP?

IS THIS BLOCK IN THE FORWARD OR FEEDBACK PATH? (F OR B)

THE CLOSED LOOP TRANSFER FUNCTION FROM THE INNER
LOOP HAS BEEN PLACED IN BLOCK NO.  2 OF THIS LOOP.

HOW MANY BLOCKS (INCLUDING BLOCK NO.  2) ARE IN
THIS LOOP?
```

In this case, the answer to the second question was 2. The system may have looked like the one below. So, the closed loop transfer function for the inner loop (enclosed by the dashed line) becomes block number 2 of the outer loop. For this system, the response to the last question would be 4.

The user is now prompted for the transfer function parameters for the remaining blocks in the outer loop (blocks 1, 3 and 4 in the example). These prompts are exactly like those presented in the Transfer Function Input section of this manual.

It is very important to note that the block number containing the inner loop's closed loop transfer function (block number 2 in the example) is always retained as a permanent parameter associated with an outer loop. This becomes important whenever an inner loop is changed and the outer loop is subsequently returned to, as discussed in the next subsection.

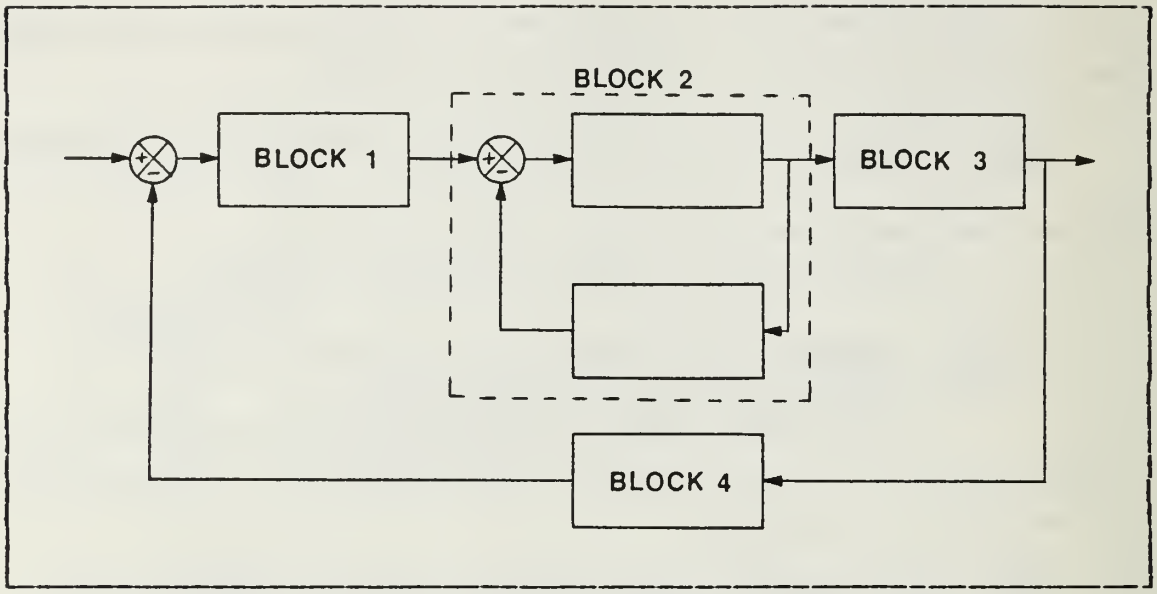


Figure A.8 Example System.

b. Entering the Outer Loop From a Data File

This option should be used whenever the user has returned to an inner loop to make changes and is now ready to re-expand to the outer loop. The outer loop must have been saved before the inner loop was loaded in order for this option to be used.

The first time that the system was expanded to the outer loop, the user specified a block in the outer loop into which the inner would be placed. That block number became a permanent parameter associated with the outer loop. For this reason, when changes have been made to the inner loop, they will automatically be reflected by a change to the correct block in the outer loop.

After the user has chosen to expand to the outer loop (using Option 4 from the Transfer Function Change menu) and defined the closed loop parameters for the inner loop, the following prompts will be presented:

```
DO YOU WISH TO INPUT THE OUTER LOOP TRANSFER FUNCTION(S)
FROM A DATA FILE OR FROM THE CONSOLE?  (D OR C)
```

WHAT IS THE NAME OF YOUR DATA FILE? (FILE NAME ONLY)

THE CLOSED LOOP TRANSFER FUNCTION FROM THE INNER LOOP HAS BEEN PLACED IN BLOCK NO. 2 OF THIS LOOP.

Naturally, one would respond with a D (data file) to the first question then provide the filename for the outer loop. The program then loads the outer loop transfer functions, but replaces the transfer function in the expansion block (block number 2 in the example) with the updated closed loop transfer function for the inner loop. A new open loop transfer function for the outer loop is also calculated using the new closed loop transfer function held in the expansion block. The user is now asked

WANT TO MAKE ANY CHANGES TO YOUR TRANSFER FUNCTION(S)? (Y OR N)

An affirmative response will send the user to the Transfer Function Change menu for the outer loop. A negative response sends the use to the DACSAP Options menu where he may select an analysis technique and observe how the changes to the inner loop have affected the total system's performance.

5. Manipulating Large Multiloop Systems

Much of the power of DACSAP is due to its ability to manipulate large multiloop systems through the use of Options 3 and 4 of the Transfer Function Change menu. The key to making these features work is to insure that the system is input starting with the innermost loop then expanding one loop at a time, saving each loop along the way. If this is done then even the innermost loop may be returned to and changed having those changes automatically incorporated in the outer loop (via Option 4 of the Transfer Function Change menu). It should be noted that the system

must be re-expanded one loop at a time when changes have been made to an inner loop. This is so each successive loop may be updated with the change to the inner loop.

The process is made particularly easy if the user simply wishes to make changes to the gains of an inner loop, which is often the case. After the inner loop is loaded, simply select Option 4 of the Transfer Function Change menu to re-expand to the outer loop. The gain parameters for the inner loop will then be presented and may be changed before the outer loop is loaded and updated.

What if an outer loop has 2 (or more) inner loops in series, as shown in Figure A.9? This situation and others

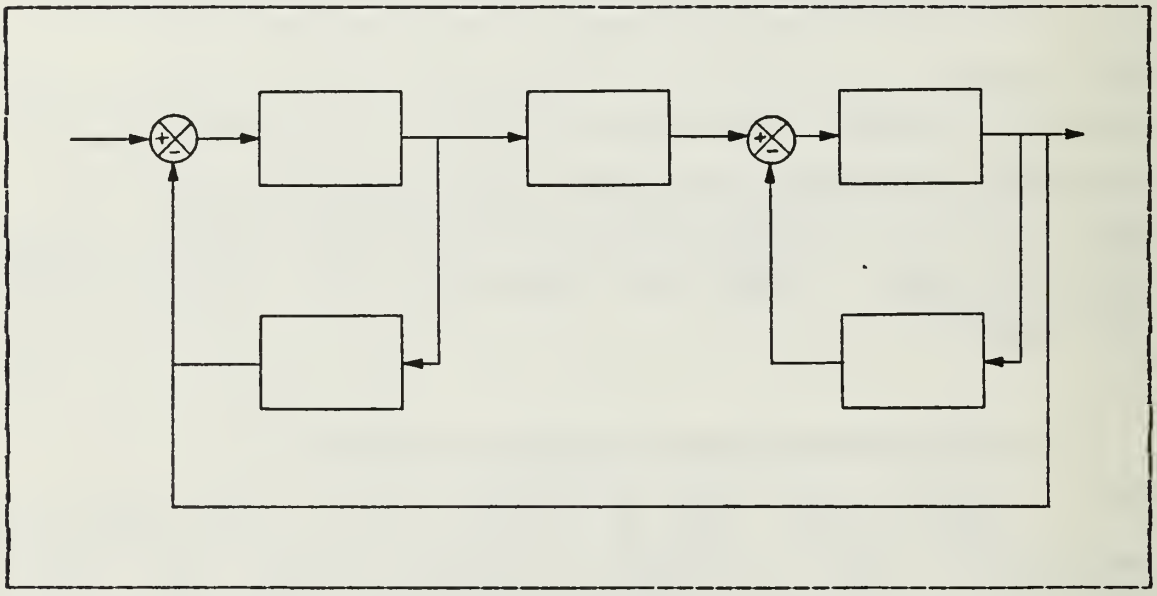


Figure A.9 Inner loops in Series.

like it can be handled by designing the inner loops as subsystems and saving them to data files. Then when designing the outer loop, these subsystems may be loaded into their appropriate blocks. If these subsystems must later be changed, then the outer loop should be saved and

the desired inner loop loaded as a new problem (Option 10 of the DACSAP Options menu). The inner loop should then be changed, re-saved and the outer loop loaded as a new problem. Then the old subsystem may be replaced by the new one through Option 5 of the Block Change menu.

The user may end a terminal session and, later, pick up where he left off if he saves the system before exiting the program. If the user has designed and saved an inner loop and, during a new terminal session, wants to begin work on the next loop (Loop 2, for example), he may do so. Recall that, during the transfer function input sequence, the user is asked for the number of the current loop. In this case, the user would respond with a 2 and would be asked for the block number in Loop 2 which contains the closed loop transfer function for Loop 1. When loading the designated block, the user will be prompted for the filename under which the inner loop (Loop 1) is saved. Changes to the inner loop may then be made using Options 3 and 4 of the Transfer Function Change menu.

One can see that loops and transfer functions may be manipulated to form systems of practically any shape. Very complex systems may be designed through the imaginative use of the subsystem block load and loop expansion features of DACSAP.

G. SAVING THE SYSTEM TO A DATA FILE

DACSAP OPTIONS	
OPTION NO.	OPTION
1	INPUT TRANSFER FUNCTION(S)
2	ROOT LOCUS ANALYSIS
3	BODE ANALYSIS (OPEN LOOP)
4	BODE ANALYSIS (CLOSED LOOP)
5	NYQUIST ANALYSIS
6	NICHOLS ANALYSIS
7	TIME RESPONSE
8	CHANGE BLOCK DIAGRAM
> 9	SAVE THIS PROBLEM <
10	START A NEW PROBLEM
11	HELP
12	EXIT DACSAP

Option 9 from the DACSAP Option menu, above, allows the user to store the system's parameters to a data file. This may be done so that the system can be loaded during a subsequent terminal session for further analysis or to continue the design process. The user will be presented with the following two questions:

```
WHAT NAME DO YOU WISH TO GIVE TO YOUR DATA FILE?
(8 CHARACTERS MAX)

DO YOU WISH TO HAVE THE CLOSED LOOP TRANSFER
FUNCTION CALCULATED AND SAVED? (Y OR N)
```

The response to the second question is important. Recall that when inputting transfer functions, any block may be loaded from a data file. The transfer function that is loaded at that time depends on which one is saved now.

If the system being saved is a feedback system, then the closed loop transfer function should be calculated and saved. Then it can be loaded as a subsystem into a block of a larger system.

If the system being saved is a compensator, filter or any other system consisting of a single component or components in cascade, then the closed loop transfer function should not be calculated. In this case, the cascade

transfer function will be saved and may be loaded into a block of another system. Note that the system will be saved on the user's 'A' disk as

'FILENAME' DATA A1

where 'FILENAME' is the name specified by the user.

H. THE HELP ROUTINE

A HELP routine exists in DACSAP which contains useful information pertaining to all of the program's major routines. These HELP files contain a list of required input data, brief operating instructions and a brief description of options for each routine. HELP may be selected from DACSAP's main option menu, the transfer function input and change menus, and each of the analysis routines' option menus. If HELP is selected from DACSAP's main option menu, any of the HELP files may be examined or the entire HELP routine may be printed out. If HELP is selected from any other option menu, information pertaining only to that routine is presented, then the user is returned to the menu from which HELP was called.

APPENDIX B

HELP ROUTINE PRINTOUT

*** DIGITAL & ANALOG CONTROL SYSTEM ANALYSIS PROGRAM ***
(DACSAP)

DACSAP IS AN INTERACTIVE COMPUTER AID FOR THE DESIGN AND ANALYSIS OF SINGLE INPUT / SINGLE OUTPUT CONTROL SYSTEMS. CLASSICAL CONTROL SYSTEM ANALYSIS METHODS ARE AVAILABLE IN THREE (3) MAIN AREAS: ROOT LOCUS, FREQUENCY RESPONSE, AND TIME RESPONSE.

CONTINUOUS SYSTEMS (S DOMAIN) AND DISCRETE SYSTEMS (Z,W, OR W' DOMAINS) MAY BE ANALYZED.

LARGE, MULTI-LOOP FEEDBACK CONTROL SYSTEMS MAY BE INPUT AND ANALYZED. EACH LOOP IS SAVED SO THAT IT MAY BE RETURNED TO AND CHANGED. ESTABLISHED (PREVIOUSLY SAVED) SUBSYSTEMS MAY BE LOADED INTO ANY BLOCK OF ANOTHER CONTROL SYSTEM. IN THIS MANNER, FEEDBACK CONTROL SYSTEMS OF NEARLY ANY SHAPE OR SIZE MAY BE GENERATED AND ANALYZED.

THE PRIMARY OUTPUT OF DACSAP IS GRAPHICAL. THEREFORE, THE PROGRAM SHOULD BE RUN FROM A GRAPHICS TERMINAL (TEK 618, TEK 4113, OR TEK 4114).

*** DACSAP TRANSFER FUNCTION INPUT ROUTINE ***

PURPOSE:

THIS ROUTINE INPUTS THE TRANSFER FUNCTION DATA REQUIRED FOR ALL OF DACSAPS ANALYSIS ROUTINES. THE INPUT DATA MUST BE AN OPEN-LOOP TRANSFER FUNCTION WHICH WILL BE INPUT AS A RATIO OF POLYNOMIALS DESCRIBED BY EITHER POLYNOMIAL COEFFICIENTS OR POLYNOMIAL ROOTS (FACTORED FORM).

REQUIRED DATA:

1. TRANSFER FUNCTION(S) WHOSE POLYNOMIALS ARE IN FACTORED OR COEFFICIENT FORM.
2. COEFFICIENTS MUST BE REAL NUMBERS ENTERED IN DESCENDING POWERS OF S, Z, W OR W'.
3. IN FACTORED FORM, THE COMPLEX ROOTS OF THE POLYNOMIAL MUST BE ENTERED ALONG WITH A REAL GAIN CONSTANT (MULTIPLICATION FACTOR).

PROCEDURE:

1. CCEFFICIENT FORM;

   GIVEN THE POLYNOMIAL

   2 S**2 + 5.656 S + 8

   THE COEFFICIENTS WOULD BE ENTERED IN THE FOLLOWING ORDER

   2
   5.656
   8

   A PROMPT WILL APPEAR BEFORE EACH ENTRY.

2. FACTORED FORM;

   THE SAME POLYNOMIAL MAY BE REPRESENTED AS

   2 (S + 1.414 + 1.414J) (S + 1.414 - 1.414J)

   AGAIN, THE USER WILL BE PROMPTED. INPUT DATA WILL BE

   CONSTANT = 2
   ROOT 1: REAL PART = -1.414
   IMAGINARY PART = -1.414
   ROOT 2: REAL PART = -1.414
   IMAGINARY PART = 1.414

3. THE FOLLOWING RESTRICTIONS APPLY TO THE TRANSFER FUNCTIONS;

   A. UP TO 20 BLOCKS MAY BE ENTERED INTO THE CONTROL LOOP.
   B. EACH BLOCK MAY BE UP TO 20TH ORDER.
   C. THE SIZE OF THE TOTAL SYSTEM MUST NOT EXCEED 100TH ORDER.
   D. THE NUMERATOR AND DENOMINATOR OF ANY GIVEN BLOCK MUST BE OF THE SAME FORM (COEFFICIENT OR FACTORED).
   E. IF ONLY ONE BLOCK IS ENTERED, IT IS ASSUMED TO BE IN THE FORWARD PATH FOR CLOSED-LOOP CALCULATIONS.
   F. UNITY FEEDBACK IS ASSUMED IF NO BLOCKS ARE ASSIGNED TO THE FEEDBACK PATH.

OPTIONS / COMMENTS:

1. ANALYSIS MAY BE PERFORMED IN THE S, Z, W, OR W' DOMAINS.
2. ANY TRANSFER FUNCTION MAY BE CHANGED FROM ANY OF THE ANALYSIS ROUTINES OR THE MAIN MENU.
3. IN ORDER TO CHANGE THE FORM (COEFFICIENT OR FACTORED) OF A BLOCK, THE USER MUST CREATE BOTH A NEW NUMERATOR AND DENOMINATOR FOR THAT BLOCK.
4. AN ENTIRE CONTROL LOOP MAY BE LOADED FROM A FILE IF IT WAS SAVED DURING A PREVIOUS TERMINAL SESSION.
5. WHEN A SYSTEM OF TRANSFER FUNCTONS IS SAVED, THE CLOSED LOOP TRANSFER FUNCTION FOR THAT SYSTEM IS ALSO SAVED.
6. THAT CLOSED LOOP TRANSFER FUNCTION CAN, THEN, BE LOADED INTO ANY BLOCK OF ANOTHER CONTROL SYSTEM. THIS ALLOWS THE USER TO DESIGN AND ANALYZE CONTROL SYSTEMS OF ALMOST ANY SIZE AND SHAPE.
7. REFER TO THE TRANSFER FUNCTION CHANGE ROUTINE HELP SECTION FOR MORE INFORMATION.

*** DACSAP TRANSFER FUNCTION CHANGE ROUTINE ***

PURPOSE:

THIS ROUTINE PROVIDES THE USER WITH THE ABILITY TO EXAMINE AND / OR CHANGE TRANSFER FUNCTIONS IN THE CONTROL LOOP CURRENTLY BEING DESIGNED OR ANALYZED. THE ROUTINE ALSO ALLOWS THE USER TO EXPAND TO OUTER LOOPS OR RETURN TO AND CHANGE INNER LOOPS.

REQUIRED DATA:

1. POLYNOMIAL(S) WHICH ARE TO BE CHANGED OR ADDED WILL BE INPUT IN THE SAME MANNER AS DESCRIBED IN THE TRANSFER FUNCTION INPUT SECTION.
2. IF EXPANDING TO OUTER LOOPS, THE INNER LOOP DATA MUST BE SAVED IF THE USER EVER INTENDS ON RETURNING TO AND CHANGING ANYTHING IN THE INNER LOOP. THE USER WILL BE PROMPTED FOR A FILE NAME DURING THIS PROCEDURE.
3. FORWARD AND FEEDBACK PATH GAINS AND THE TYPE OF FEEDBACK ARE REQUIRED WHEN LOOPS ARE SAVED. THIS DATA IS NEEDED IN ORDER TO CALCULATE THE LOOPS CLOSED LOOP TRANSFER FUNCTION.

PROCEDURE:

1. SELECT THE DESIRED OPERATION FROM THE TRANSFER FUNCTION CHANGE MENU.
2. THE USER WILL BE PROMPTED WHILE MAKING CHANGES TO ANY BLOCK IN THE CURRENT LOOP.
3. IF THE FORM OF AN EXISTING BLOCK IS TO BE CHANGED, THE USER MUST SELECT THE OPTION TO CREATE BOTH A NEW NUMERATOR AND DENOMINATOR FOR THAT BLOCK.
4. WHEN RE-ENTERING INNER LOOPS, SAVE THE CURRENT LOOP AND PROVIDE THE FILE NAME OF THE LOOP TO BE RE-EXAMINED. EXPANDING BACK OUT TO THE OUTER-MOST LOOP MUST BE DONE ONE LOOP AT A TIME. THIS WILL ALLOW THE PROGRAM TO AUTOMATICALLY UPDATE THE OUTER LOOPS WITH THE CHANGES MADE IN THE INNER LOOP.
5. SAVE EACH LOOP DURING THE EXPANSION PROCESS SO THAT THE LOOP FILES CONTAINS THE MOST UP-TO-DATE INFORMATION.

OPTIONS / COMMENTS:

1. EXISTING TRANSFER FUNCTIONS MAY BE CHANGED OR MOVED.
2. NEW TRANSFER FUNCTION(S) MAY BE ADDED TO THE CONTROL LOOP.
3. THE CONTROL SYSTEM MAY BE EXPANDED TO MULTIPLE LOOPS. THE INNER LOOP CLOSED LOOP TRANSFER FUNCTION IS AUTOMATICALLY PLACED INTO A USER DEFINED BLOCK IN THE OUTER LOOP.
4. INNER LOOPS WHICH HAVE BEEN SAVED MAY BE LOADED AND CHANGED. ONCE CHANGES HAVE BEEN MADE, THE OUTER LOOPS CAN AUTOMATICALLY BE UPDATED IF EXPANSION IS DONE ONE LOOP AT A TIME.

*** DACSAP ROOT LOCUS ROUTINE ***

PURPOSE:

THIS ROUTINE CALCULATES THE SYSTEMS CLOSED LOOP CHARACTERISTIC EQUATION FROM A PREVIOUSLY ENTERED OPEN LOOP TRANSFER FUNCTION. THE ROOTS OF THE EQUATION ARE DETERMINED OVER A USER DEFINED RANGE OF OPEN LOOP GAIN VALUES AND MAY BE PLOTTED AND/OR PRESENTED IN TABULAR FORM.

REQUIRED DATA:

1. A PREVIOUSLY ENTERED OPEN LOOP TRANSFER FUNCTION.
2. A RANGE OF OPEN LOOP GAIN VALUES. THESE GAINS MAY BE POSITIVE OR NEGATIVE, BUT NOT BOTH.
3. TYPE FEEDBACK AND SAMPLE PERIOD (DISCRETE SYSTEMS).
4. PLOTTING DIMENSIONS. RECTANGULAR DIMENSIONS (IE. XMAX-XMIN = YMAX-YMIN) SHOULD BE USED IF ANGLES OR MAGNITUDES WILL BE MEASURED DIRECTLY FROM THE PLOT. IF UNSURE OF THE PLOTTING DIMENSIONS, SELECT TABULAR DATA FIRST. POLE AND ZERO LOCATIONS WILL BE PROVIDED SO THAT PLOTTING DIMENSIONS MAY BE DETERMINED.

PROCEDURE:

1. SELECT PLOTTED OR TABULAR DATA OPTION.
2. INPUT GAIN AND PLOTTING PARAMETERS WHEN PROMPTED.
3. AFTER THE ROOT-LOCUS HAS BEEN PLOTTED, THE GAIN, NATURAL FREQUENCY, AND DAMPING RATIO FOR SPECIFIC POINTS ON THE LOCUS MAY BE REQUESTED. THEN, THE ROOTS ASSOCIATED WITH ANY GAIN VALUE MAY BE HIGHLIGHTED ON THE PLOT.
4. TABULAR DATA MAY BE OBTAINED AFTER PLOTTING.

OPTIONS / COMMENTS:

1. TEMPLATES FOR CONSTANT DAMPING RATIO AND NATURAL FREQUENCY ARE PROVIDED FOR DISCRETE DOMAIN PLOTS.
2. GAIN, DAMPING RATIO, AND NATURAL FREQUENCY MAY BE REQUESTED FOR ANY POINT ON THE ROOT-LOCUS.
3. A DESIGN POINT MAY BE HIGHLIGHTED ON THE PLOT BY SPECIFYING ITS GAIN VALUE.
4. 0-4 LINES OF HEADING MAY BE SPECIFIED FOR EACH PLOT.

*** DACSAP FREQUENCY RESPONSE ROUTINE ***

PURPOSE:

THIS ROUTINE CALCULATES AND PLOTS THE FREQUENCY RESPONSE OF AN OPEN OR CLOSED LOOP SYSTEM TO A USER SPECIFIED INPUT SIGNAL AND FREQUENCY RANGE. OPEN LOOP RESPONSE MAY BE ANALYZED USING BODE PLOTS, POLAR PLOTS (NYQUIST ANALYSIS), OR PHASE-MAGNITUDE PLOTS (NICHOLS ANALYSIS). CLOSED LOOP RESPONSE MAY BE ANALYZED USING BODE PLOTS.

REQUIRED DATA:

1. A PREVIOUSLY ENTERED OPEN LOOP TRANSFER FUNCTION.
2. OPEN LOOP GAIN (BROKEN INTO FORWARD AND FEEDBACK PATH GAINS FOR CLOSED LOOP RESPONSE).
3. RANGE OF FREQUENCIES.
4. SAMPLE PERIOD (DISCRETE SYSTEMS).
5. TYPE OF FEEDBACK (DISCRETE SYSTEMS)

PROCEDURE:

1. SELECT PLOTTED OR TABULATED DATA OPTION.
2. INPUT THE REQUIRED DATA (SEE ABOVE) WHEN PROMPTED.
3. ALL PLOTS ARE AUTOMATICALLY SCALED. FOR NYQUIST PLOTS, IF A LARGE FREQUENCY RANGE IS SELECTED, THE AUTOMATIC SCALING CAN RESULT IN LOW RESOLUTION NEAR THE ORIGIN. INCREASING THE MIN FREQ MAY RESULT IN A BETTER PLOT.
4. TABULAR DATA MAY BE OBTAINED AFTER PLOTTING.

OPTIONS / COMMENTS:

1. OPEN LOOP PHASE AND GAIN MARGINS AND CROSS-OVER FREQUENCIES ARE PROVIDED AFTER EACH PLOT.
2. FOR THE DISCRETE DOMAINS, FREQUENCY WARPING OCCURS DURING THE CALCULATIONS. THE USER SPECIFIES WHETHER THE RESPONSE IS TO BE PLOTTED AGAINST THE FICTITIOUS (WARPED) FREQUENCIES OR REAL FREQUENCIES.
3. 0-4 LINES OF HEADING MAY BE SPECIFIED FOR EACH PLOT.

*** DACSAP TIME RESPCNSE ROUTINE ***

PURPOSE:

THIS ROUTINE CALCULATES AND PLCTS THE TRANSIENT RESPONSE OF THE CLOSED LOOP SYSTEM TO A USER SPECIFIED INPUT. THE ROUTINE CALCULATES THE CLOSED LOOP TRANSFER FUNCTION FROM OPEN LOOP DATA AND INPUTS TO THIS ROUTINE.

REQUIRED DATA:

1. A PREVIOUSLY ENTERED OPEN LCOP TRANSFER FUNCTION.
2. TYPE AND MAGNITUDE OF THE INPUT SIGNAL.
3. FORWARD AND FEEDBACK PATH GAINS AND THE TYPE OF FEEDBACK.
4. SAMPLE PERIOD (DISCRETE SYSTEMS).
5. THE NUMBER OF SECCNDS OVER WHICH THE RESPONSE IS TO BE OBSERVED.

PROCEDURE:

1. SELECT PLOTTED OR TABULATED DATA OPTION.
2. ENTER THE REQUIRED DATA (SEE ABOVE) WHEN PROMPTED.
3. TABULAR DATA MAY BE OBTAINED AFTER PLOTTING.

OPTIONS / COMMENTS:

1. TRANSIENT RESPONSE MAY BE OBTAINED FOR IMPULSE, STEP, OR RAMP INPUTS.
2. A CONTINUOUS OUTPUT IS PLOTTED FOR LAPLACE TRANSFER FUNCTIONS. FOR DISCRETE SYSTEMS, THE RESPONSE IS PLOTTED ONLY AT INTERVALS OF THE SAMPLE PERIOD.
3. 0-4 LINES OF HEADING MAY BE SPECIFIED FOR EACH PLOT.

APPENDIX C
DISSPLA AND GRAPHICAL OUTPUT DEVICES

The main purpose of DACSAP is to provide the user with graphical systems analysis tools such as root locus plots, frequency response plots and time response plots. DACSAP takes in user specified parameters and performs calculations which result in data which is to be plotted. The actual plotting, however, is done by a library of graphics subroutines called DISSPLA (Display Integrated Software System and Plotting Language). DISSPLA is a web of FORTRAN subroutines to which plotting parameters are passed and which, in turn, provide the instructions necessary to produce high quality plots, graphs, charts or other specialized graphics.

DACSAP passes data which is to be plotted to one of its plotting routines; RTLCS, BPLOT, NYPLOT, NICPLT, or TPLOT. These routines, then, pass the parameters to DISSPLA by calling a variety of subroutines in the proper sequence to produce the desired plot. By examining one of the plotting routines, the reader can see that, in some cases, no more than 30 instructions are required to produce a high quality plot with curve smoothing, plotting surface grid, labelled axes and headings.

In general, DISSPLA subroutine names are indicative of their function. For example, the subroutine CURVE passes to DISSPLA the array names containing the data which defines a curve. XNAME and YNAME pass the x and y axis labels, respectively. GRAF, XLOG, and POLAR define the dimensions for rectangular, semi-log and polar plotting surfaces, respectively. Most of DISSPLA's subroutines follow this pattern, thus, making the graphics routines very easy to implement. Even if the user is unfamiliar with DISSPLA, he

could examine one of DACSAP's plotting routines and follow the method used to produce a plot.

In general, DACSAP controls the size, shape, scaling and nature of information presented on all plots. Thus, the user is faced with far fewer questions from the program, but because all plotting decisions are made by the program, he has little control over the look of the plot. The only parameter controlled by the user is the plot heading. DISSPLA allows up to four lines of heading per plotted page. DACSAP accommodates this capability, restricting the length of each line to 32 characters.

The orientation of root locus, polar and magnitude-phase frequency response plots are changed based on whether a heading is selected or not. These plots are presented sideways on the TEK 618 when a heading is selected so that the long side of the plot is aligned with the long edge of the screen. In this way, hardcopy prints of the screen better fill the page. All plots are presented upright if no heading is selected. If the user intends on running a multitude of plots, conducting analysis on the screen, it is recommended that the plots be done without headings. This reduces the plotting time and presents the plots upright. A heading may be added at any time and the plot regenerated with a heading if a hardcopy is desired.

DISSPLA produces plots on a number of output devices. The user of DACSAP is given four output device options; TEK 618, TEK 4113/4114, IBM 3278 and DISSPLA metafile. Since DACSAP is meant to be an interactive design tool, it is best utilized when output is to a high speed, high resolution device such as the TEK 618 or TEK 4113/4114, using a high speed modem. The TEK 4113/4114 may be too slow for productive interactive design if used with baud rates of 1200 or less.

Output to the IBM 3278 is in the form of a low resolution, unlabelled, line printer style plot. The general shape of curves can be observed, but real analysis of these plots is practically impossible. However, this option may be useful if the user wishes to input the transfer functions for a large system, make changes to a system or obtain tabular output, and do so without tying up a graphics terminal.

The option to output to a metafile is available to those users who wish to create publication quality plots using a postprocessor and high resolution plotting device such as a VERSATEC electrostatic plotter. The plots, in this case, are sized so as to fit comfortably on an 8.5 by 11 inch sheet of paper. All figures in this thesis were produced using the metafile option. This option may not be used for analysis, however, since output does not appear at the terminal. Separate terminal sessions are required; the first to analyze the system and the second to produce plots based on parameters obtained in the analysis session.

APPENDIX D
PROGRAMMER'S GUIDE

This appendix contains detailed information about each of DACSAP's subroutines. The main purpose of this appendix is to provide supplemental information to that found in Chapter 3 of this report. Chapter 3 describes the capabilities and options available from the major routines and outlines some of the equations used. This guide outlines the specific functions of the driving program, DACSAP, and each of the subroutines. DACSAP is presented first, then the subroutines are presented in alphabetical order. The outline for each contains definitions of the main variables and arrays, notes about the routine's operation and a list of subprograms called by that routine. Table I is a cross reference of subprograms showing the calling routines for each subprogram.

It is hoped that the combination of the program description found in Chapter 3 and the more detailed subroutine descriptions found in this appendix will help anyone interested in interpreting the actual FORTRAN code for the program.

D.1 DACSAP

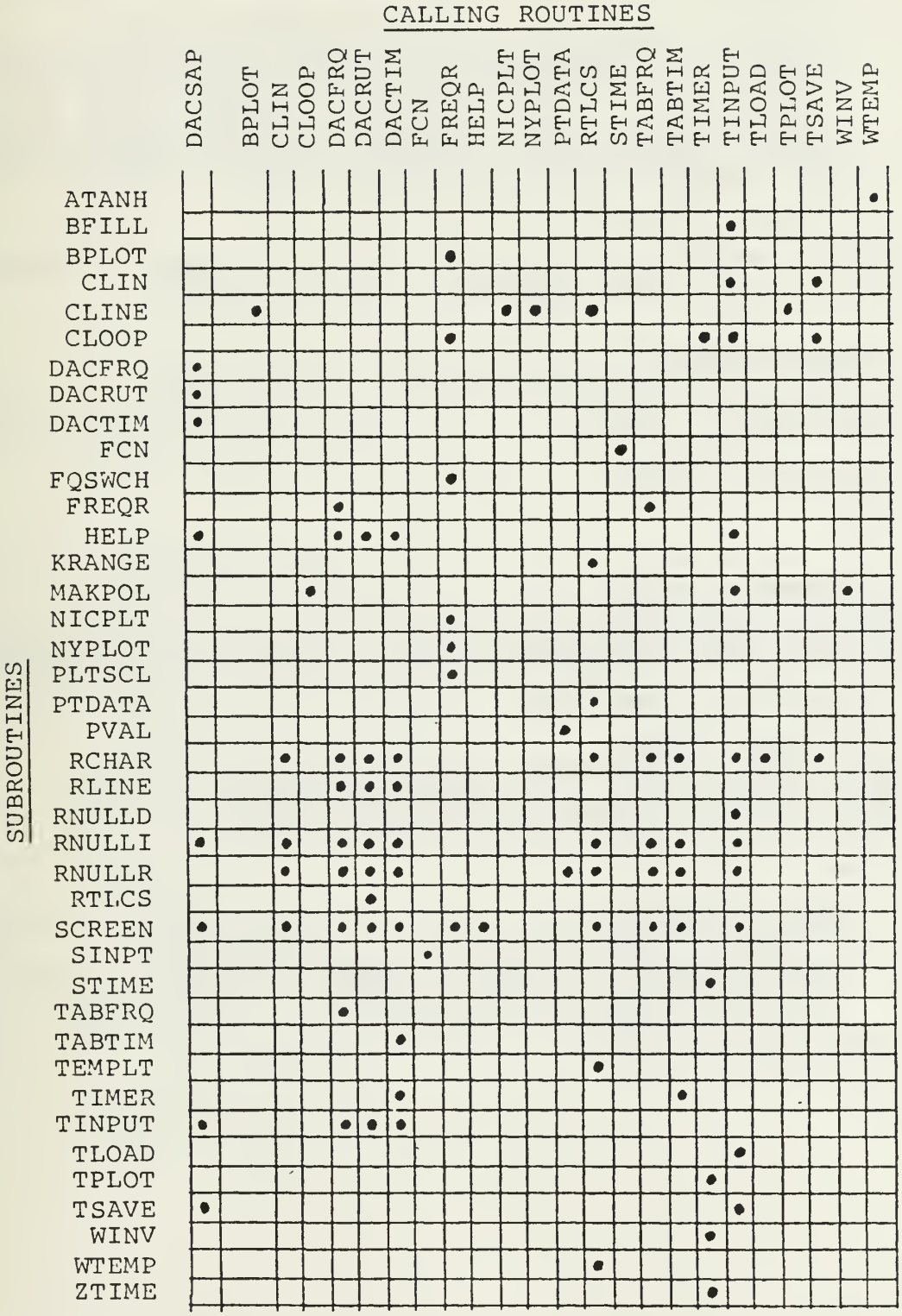
DACSAP is the driving program which presents the main option menu and calls subroutines based on the user's selection from that menu.

Definition of variables

IRS = Problem restart flag
ITRM = Graphical output device number
FD = Printed output device number
TINIT = Transfer function parameter initialization flag
FINIT = Frequency response parameter initialization flag
TMINIT = Time response parameter initialization flag
RINIT = Root locus parameter initialization flag

TABLE I

Subroutine Cross Reference



Notes

1) The routine insures that the transfer function parameters have been defined before any analysis routines are called by checking the condition of TINIT.

2) Analysis routine parameter initialization flags (RINIT, FINIT,TMINIT) are used to determine where, within the called subroutine, control will be transferred. If the flag is 0, control will go to the parameter input rcutine. If the flag is 1, control will go to the parameter change routine. Values greater than 1 are temporarily used to indicate special event sequences.

Subprograms called

DACFRQ, DACRUT, DACTIM, HEIP, RNULLI, SCREEN, TINPUT, TSAVE DISSPLA subroutines.

D.2 FUNCTION ATANH

This function evaluates the inverse hyperbolic tangent of a real number, $-1<x<1$.

Subprograms called

None

D.3 SUBROUTINE BFILL

This subroutine fills a block with the closed loop transfer function of another system.

Definition of variables

BN = Block number
NNM = Order of the numeratcr polynomial
NDN = Order of the denominator polynomial
BNUM = Array of numerator coefficients
BDEN = Array of denominator coefficients

Notes

1) See Subroutine CLOOP for a definition of closed loop parameters.

2) This routine is called anytime the user chooses to input a block's transfer function from a data file or when expanding the system to an outer loop.

Subprograms called

None

D.4 SUBROUTINE BPLOT

This subroutine contains all of the calls to DISSPLA subroutines necessary to create a Bode frequency response plot.

Definition of variables

WMIN = Low end of frequency range
WMAX = High end of frequency range
AMIN = Minimum amplitude
WFREQ = Array of frequencies to be plotted
QMAG = Array of magnitudes (dB) to be plotted
QPHSD = Array of phase values (deg) to be plotted
LINES = Array of plot heading character strings

Notes

1) Subroutine PLTSCL is called prior to this one to calculate plot dimensions based on the values in the arrays QPHSD and QMAG.

2) The routine plots both magnitude and phase on the same plot with magnitude labelled on the left side of the plot and phase labelled on the right.

Subprograms called

CLINE, DISSPLA subroutines

D.5 SUBROUTINE CLIN

CLIN is the subroutine used to prompt the user for the closed loop parameters and is used whenever the program is going to save the closed loop transfer function or going to calculate the closed loop transfer function of an inner loop prior to a system expansion.

Definition of variables

KG = Forward path gain
KH = Feedback path gain
TYPFB = Type of feedback (P=positive, N=negative)
II = Flag which determines how user will be prompted

Notes

1) This routine checks to see if the closed loop parameters have been previously defined by some other routine. If so, the user is simply presented with the currently held values and given the opportunity to change them, if so desired.

2) The subroutine has an integral change and correction routine.

Subprograms called

RCHAR, RNULLI, RNULLR, SCREEN

D.6 SUBROUTINE CLINE

This routine places one row of the 2-D array LINES into a 1-D array suitable for use by DISSPLA subroutines.

Definition of variables

ILN = Line number (row number of LINES)
LINES = 2-D array of plot heading lines
LINE = 1-D array containing one row of LINES

Notes

1) The dollar sign delimiter is placed at the end of the character string as required by DISSPLA.

Subprograms called

None

D.7 SUBROUTINE CLOOP

This subroutine calculates the coefficients of the system's closed loop transfer function. The inputs for the routine are the system's open loop transfer function and the individual block transfer functions.

Definition of variables

CLNN = Order of the closed loop numerator
CLND = Order of the closed loop denominator
CNM = Array of closed loop numerator coefficients
CDEN = Array of closed loop denominator coefficients
CLRTN = Array of closed locp numerator roots
KN = Closed loop numerator gain constant
NNUM = Order of the open loop numerator
NDEN = Order of the open loop denominator
NM = Array of open loop numerator coefficients
DEN = Array of open loop denominator coefficients
RTN = Array of open loop numerator roots
RTD = Array of open loop denominator roots
KG = Forward path gain
KH = Feedback path gain
TFID = Array of transfer function I.D. numbers (see TINPUT)

Notes

1) The following equation was developed, using Mason's gain rule, for calculating the closed loop transfer function, CLTF:

$$CLTF = \frac{KG \times NG \times DH}{(DG \times DH) + (KG \times KH \times NG \times NH)}$$

$$= \frac{KG \times NG \times DH}{DGH + (KG \times KH \times NGH)}$$

where NG = forward path numerator
NH = feedback path numerator
DG = forward path denominator
DH = feedback path denominator
NGH = open loop numerator
DGH = open loop denominator

2) Multiplication of NG x DH is done by combining their roots, then calculating the resulting polynomial coefficients.

Subprograms called

MAKPOL

D.8 SUBROUTINE DACFRQ

This subroutine is the frequency response parameter input routine. It creates all prompts and menus required to interactively input and change frequency response analysis parameters.

Definition of variables

KG = Forward path gain
KH = Feedback path gain
TYPFB = Type of feedback (P=positive, N=negative)
K = Open loop gain
TYFQ = Type of frequency (w=real, v=ficticious)
T = Sample period
WMIN = Low end of frequency range
WMAX = High end of frequency range
WNY = Nyquist frequency (discrete systems)
NL = Number of lines in the plot heading
LINES = Array of plot heading character strings
TYPBOD = Type Bode flag (0=open loop, 1=closed loop)
TYPP = Type of plot flag
CAC = Change and correction flag (same as parameter initialization flag, see DACSAP)
RN = Return number, determines where control should return to after correction routine
SN = Screen number
M1 = Change menu flag
M = Parameter change flag
MM = Parameter correction flag

Notes

1) Open or closed loop parameters are input based on the value of TYPBOD.

2) Corrections routine is used to display to the user the parameters which have just been entered. Commanded GO TO statements are used to select the correct line number in the input routine when corrections are necessary.

3) Change routine calls the parameter change menu, then selects the corrections routine if parameters are to be changed. Subroutine TINPUT is called if changes to transfer functions are desired.

4) When no further changes are desired, the calculation routine FREQR is called.

Subprograms called

FREQR, HELP, RCHAR, RLINE, RNULLI, RNULLR, SCREEN, TABFRQ, TINPUT

D.9 SUBROUTINE DACROT

This subroutine is the root locus parameter input routine. It creates all prompts and menus required to interactively input and change root locus analysis parameters.

Definition of variables

KMIN = Minimum open loop gain value
KMAX = Maximimum open loop gain value
TYPFB = Type of feedback (P=positive, N=negative)
ITIC = Tic mark flag
KTICI = Gain increment for tic marks
T = Sample period (discrete systems)
XMAX,XMIN = x-axis dimensions
YMAX,YMIN = y-axis dimensions
NL = Number of lines in the plot heading
LINES = Array of plot heading character strings
CAC = Change and correction flag (same as parameter initialization flag, see DACSAP)
SN = Screen number
M1 = Change menu flag
M = Parameter change flag
MM = Parameter correction flag

Notes

1) Value of ITIC determines whether tic marks will be placed on the root locus plot.

2) Corrections routine is used to display to the user the parameters which have just been entered. Commanded GO TO statements are used to select the correct line number in the input routine when corrections are necessary.

3) Change routine calls the parameter change menu, then selects the corrections routine if parameters are to be changed. Subroutine TINPUT is called if changes to transfer functions are desired.

4) When no further changes are desired, the calculation routine RTLCS is called.

Subprograms called

HELP, RCHAR, RLINE, RNULLI, RNULLR, RTLCS, SCREEN, TINPUT

D.10 SUBROUTINE DACTIM

This subroutine is the time response parameter input routine. It creates all prompts and menus required to interactively input and change time response analysis parameters.

Definition of variables

IN = Type input designator
UMAX = Max amplitude of input
KG = Forward path gain
KH = Feedback path gain
TYPFB = Type of feedback (P=positive, N=negative)
T = Sample period (discrete systems)
TMAX = Length of time over which response is to be analyzed
NL = Number of lines in the plot heading
LINES = Array of plot heading character strings
CAC = Change and correction flag (same as parameter initialization flag, see DACSAP)
RN = Return number, determines where control should return to after correction routine
SN = Screen number
M1 = Change menu flag
M = Parameter change flag
MM = Parameter correction flag

Notes

1) The prompts used depend on whether the system is continuous or discrete.

2) Corrections routine is used to display to the user the parameters which have just been entered. Commanded GO TO statements are used to select the correct line number in the input routine when corrections are necessary.

3) Change routine calls the parameter change menu, then selects the corrections routine if parameters are to be changed. Subroutine TINPUT is called if changes to transfer functions are desired.

4) When no further changes are desired, the calculation routine TIMER is called.

Subprograms called

HELP, RCHAR, RLINE, RNULLI, RNULLR, SCREEN, TABTIM, TIMER, TINPUT

D.11 SUBROUTINE FCN

This subroutine defines the canonical form of the state equations, based on the closed loop transfer function, which are used to calculate the s-domain time response.

Definition of variables

XDOT = Array of values for each differential equation at time, ITIME
ITIME = Time values at which calculations occur
X = Array of the current values of each state
MN = Number of states
UIN = Input amplitude at time, ITIME

Notes

1) This routine is called by the IMSL differential equation solving routine DVERK.

Subprograms called

SINPT

D.12 SUBROUTINE FQSWCH

This subroutine performs the bilinear frequency transformation for frequency response analysis in the w- or w'-domains.

Definition of variables

FQF = Type-of-frequency-transform designator
NOF = Number of frequency value in the array WFREQ
WFREQ = Array of frequency values to be plotted
T = Sample period

Notes

1) The value of FQF is set by the calling routine, FREQR, and determines in which domain and in which direction the transform is being performed.

Subprograms called

None

D.13 SUBROUTINE FREQR

This subroutine is the main calculation routine for frequency response analysis. Its purpose is to evaluate the appropriate transfer function at $j\omega$, $j\nu$ or $j\nu'$, depending on the domain.

Definition of variables

TYPBOD = Type Bode flag (0=open loop, 1=closed loop)
TYPP = Type of plot flag
NCOF = Array of numerator coefficients
DCOF = Array of denominator coefficients
NOF = Number of frequencies in array WFREQ
WFREQ = Array of frequencies for evaluation
QNUMR,QNUMI = Real and imaginary value of the numerator at $j\omega$
QDENR,QDENI = Real and imaginary value of the denominator at $j\omega$
QMAG = Magnitude of the transfer function at $j\omega$
QR,QI = Real and imaginary part of QMAG
QPHSR = Array of phase values (radians)
QPHSD = Array of phase values (degrees)
PSF = Phase shift flag
IDOM = Domain of transfer functions
PM = Phase margin
GM = Gain margin
PMF = Phase margin flag
GMF = Gain margin flag

Notes

1) Subroutine PVAL is called to evaluate the numerator and denominator polynomials at $j\omega$, $j\nu$ or $j\nu'$.

2) Complex division is perfomed to determine the values of QR and QI.

3) QR and QI are evaluated to determine phase values.

4) TYPBOD determines whether NCOF and DCOF are coefficients of the open loop or closed loop transfer function.

5) If the frequency range chosen by the user allows, the gain and phase margins are calculated. PMF and GMF are set if gain and phase crossovers occur. The condition of these flags determines whether PM and GM are evaluated.

6) The value of TYPP determines what type of plot will be used to depict the frequency response.

Subprograms called

BPLOT, CLOOP, FQSWCH, NICPIT, NYPLOT, PLTSCL, SCREEN

D.14 SUBROUTINE HELP

This subroutine provides information about the operation of each of DACSAP's major subroutines. The routine may be accessed from any of the the program's option menus.

Definition of variables

I = Calling routine designator
FD = Output device number

Notes

1) The routine for which information is to be provided is determined by the value of I.

Subprograms called

SCREEN

D.15 SUBROUTINE KRANGE

This subroutine determines the number of roots of a polynomial which are out of the plotting area of a root locus plot.

Definition of variables

N = Order of the polynomial being evaluated
A = Array of coefficients of the characteristic polynomial
XMIN, XMAX = x-axis dimensions of the root locus plot
YMAX = y-axis max dimension
Z = Array of complex roots of polynomial, A
X, Y = Real and imaginary parts of Z
OUT = Number of roots which are outside the plotting area

Notes

1) The routine is called by RTLCS to determine if the user's range of gain values is too large for the chosen plotting area.

Subprograms called

None

D.16 SUBROUTINE MAKPOL

This subroutine calculates the complex coefficients of a polynomial given the roots of that polynomial.

Definition of variables

N = Order of the polynomial
R = Array of complex roots of the polynomial
C = Array of complex coefficients of the polynomial

Notes

1) The value of the coefficient of the highest power of the unknown is always unity and is not returned by the routine.

Subprograms called

None

D.17 SUBROUTINE NICPLT

This subroutine contains all of the calls to DISSPLA subroutines necessary to create a log magnitude-phase plot for use in Nichols frequency response analysis.

Definition of variables

YORIG,YMX = Dimensions of the phase axis
AMIN,AMAX = Dimensions of the magnitude axis
QPHSD = Array of frequencies to be plotted
QMAG = Array of magnitude to be plotted
LINES = Array of plot heading character strings

Notes

1) Subroutine PLTSCL is called prior to this one to calculate the plot dimensions based on values in the arrays QPHSD and QMAG.

2) The plot is rotated on its side if a heading exists. This will cause the plot to better fill the screen and cause any printout of the screen to better fill the page.

Subprograms called

CLINE

D.18 SUBROUTINE NYPLOT

This subroutine contains all of the calls to DISSPLA subroutines necessary to create a polar plot of magnitude and phase for use in Nyquist frequency response analysis.

Definition of variables

QRMIN,QRMAX = Max and min values of the array QR
QIMIN,QIMAX = Max and min values of the array QI
QM = Array of magnitude values
QR,QI = Real and imaginary part of QM
RL = Resulting length of axis
RSTEP = Axis step size
XDIST,YDIST = X and Y position of origin of polar plot
LINES = Array of plot heading character strings

Notes

1) The scaling routine is used to determine the dimensions of the plotting area based on the arrays QR and QI.

2) The plot is rotated on its side if a heading exists. This will cause the plot to better fill the screen and cause any printout of the screen to better fill the page.

Subprograms called

CLINE

D.19 SUBROUTINE PLTSCL

This subroutine calculates the dimensions for frequency response plots based on the arrays of values to be plotted.

Definition of variables

AMIN,AMAX = Min and max values of the array QMAG
PMIN,PMAX = Min and max values of the array QPHSD
YORIG,YMX = Min and max values of the phase axis
QMAG = Array of magnitude values (dB)
QPHSD = Array of phase values (degrees)

Notes

1) The routine insures that numbers that are easy to work with are used on the axes of frequency response plots.

Subprograms called

None

D.20 SUBROUTINE PTDATA

This subroutine calculates the open loop gain, damping ratio and natural frequency of any point on a root locus plot.

Definition of variables

X,Y = Point designated by the user
T = Sample period
SIG = Damping
WD = Damped natural frequency
IDOM = Domain of the transfer functions
D = Damping ratio
W = Undamped natural frequency
K = Open loop gain
XD,YD = Comparison x and y values
TOL = Tolerance value for reading X and Y from the root locus plot

Notes

1) The routine solves explicitly for values in the s-, w- or w'-domains.

2) w- and w'-plane natural frequency, W, can be expressed as

$$W = (SIG^2 + WD^2)^{1/2}$$

3) w- and w'-plane damping ratio, D, can be expressed as

$$D = | SIG / W |$$

4) w- and w'-plane damping and damped natural frequency are

w-plane	w'-plane
$SIG = (2/T) \tanh^{-1} X$	$SIG = (2/T) \tanh^{-1} (X\ T/2)$
$WD = (2/T) \tan^{-1} Y$	$WD = (2/T) \tan^{-1} (Y\ T/2)$

5) The z-plane iteration routine calculates X and Y values for all values of damping ratio and natural frequency, compares them to the user's X and Y value and displays the values of W and D which correspond to the closest match.

Subprograms called

None

D.21 SUBROUTINE PVAL

This subroutine evaluates a polynomial at any complex value of the variable.

Definition of variables

A = Array of polynomial coefficients
NN = Order of the polynomial, A
S = Complex value at which the polynomial is being evaluated
P = Complex value of the polynomial at S
PR,PI = Real and imaginary parts of S
VR,VI = Real and imaginary parts of P

Subprograms called

None

D.22 SUBROUTINE RCHAR

This subroutine is used to read a literal reply to an interactive prompt allowing for a recovery from an inadvertent null string entry.

Definition of variables

IANS = Character response to the prompt
COUNT = Error count

Notes

1) Two null string entries in a row and program execution will end.

Subprograms called

None

D.23 SUBROUTINE RLINE

This subroutine reads in a character string from the console allowing for a recovery from an inadvertent null string entry.

Definition of variables

ILN = Number of lines in the array LINES
LINES = 2-D array of characters

Notes

1) Two null string entries in a row and program execution will end.

Subprograms called

None

D.24 SUBROUTINE RNULLD

This subroutine is used to read a double precision real number reply to an interactive prompt allowing for a recovery from an inadvertent null string entry.

Definition of variables

ANSR = Double precision response to the prompt
COUNT = Error count

Notes

1) Two null string entries in a row and program execution will end.

Subprograms called

None

D.25 SUBROUTINE RNULLI

This subroutine is used to read an integer reply to an interactive prompt allowing for a recovery from an inadvertent null string entry.

Definition of variables

IANS = Integer response to the prompt
COUNT = Error count

Notes

1) Two null string entries in a row and program execution will end.

Subprograms called

None

D.26 SUBROUTINE RNULLR

This subroutine is used to read a single precision real number reply to an interactive prompt allowing for a recovery from an inadvertent null string entry.

Definition of variables

ANSR = Single precision response to the prompt
COUNT = Error count

Notes

1) Two null string entries in a row and program execution will end.

Subprograms called

None

D.27 SUBROUTINE RTLCS

This subroutine is the calculation and plotting routine for root locus analysis.

Definition of variables

POUT = Number of zeroes outside the plotting area
KMX = Calculated max value of gain which will be used for plotting purposes
KLIM = Interim value of KMX
KOUNT = Number of roots plotted
NNUM = Order of the open loop numerator
NDEN = Order of the open loop denominator
NM = Array of open loop numerator coefficients
NUM = Array of open loop numerator coefficients with leading zeroes
DEN = Array of open loop denominator coefficients
Z = Array of complex polynomial roots (output of ZPOLR)
XPOLE,YPOLE = Real and imaginary parts of system poles
XZERO,YZERO = Real and imaginary parts of system zeroes
LINES = Array of plot heading character strings
XMIN,XMAX = x-axis plot dimensions
YMIN,YMAX = y-axis plot dimensions
A = Array of characteristic polynomial coefficients
K = Open loop gain used in calculating A
KP = Particular value of gain to be highlighted on plot
KTMIN,KTMAX = Range of gain values for tabulated output
KTINC = Gain increment for tabulated ouput

Notes

1) This routine is a calculation, plotting and tabular data routine. This is done only for root locus analysis because roots are calculated for only one set of roots at a time, plotted or tabulated and, then, gain is incremented and the process begins again. Arrays containing all of the roots and gain values are not formed because they would take up too much memory.

2) The plotting routine contains all of the DISSPLA subroutine calls necessary to draw a root locus plot except for the discrete domain plots. In these cases, TEMPLT or WTEMP are also called to draw the lines of constant damping ratio and natural frequency.

3) The beginning of the routine adjusts the max gain value if the user has chosen one which is so large that the roots cannot be plotted in the plotting area selected.

Subprograms called

CLINE, KRANGE, PTDATA, RCHAR, RNULLI, RNULLR, SCREEN, TEMPLT, WTEMP, ZPOLR (IMSL)

D.28 SUBROUTINE SCREEN

This subroutine contains the instructions required to display, at the terminal, all option menus and parameter change and display screens.

Definition of variables

SN = Screen number

Notes

1) A description of variables used in each of the screens may be found in the associated calling routine.

Subprograms called

None

D.29 SUBROUTINE SINPT

This subroutine calculates the magnitude of the input at any time for time response calculations.

Definition of variables

IN = Type-of-input designator
ITIME = Time at which calculation is taking place
TINT = Time interval for ITIME
TMAX = Max time (sec) over which response is observed
UMAX = Max amplitude of the input
UIN = Amplitude of the input at time ITIME

Notes

1) Variable IN is established by DACTIM and determines the type of input (step, impulse or ramp).

Subprograms called

None

D.30 SUBROUTINE STIME

This subroutine calculates the coefficients of the canonical form of the state equations based on the closed loop transfer function coefficients.

Definition of variables

TOL = Integration tolerance
TIME = Array of time values
ITIME = Time at which calculation is taking place
X = Array of state variables
CLNN = Order of the closed loop numerator
CLND = Order of the closed loop denominator
CNM = Array of closed loop numerator coefficients
CDEN = Array of closed loop denominator coefficients
CNUM = Array of closed loop numerator coefficients with leading zeroes
A = Coefficients of the companion matrix
B = Coefficients of the output equation
AMP = Array of output amplitude values

Notes

1) The array A contains the coefficients of the bottom row of the canonical form of the state matrix. These coefficients are based on the denominator coefficients of the closed loop transfer function.

2) The remainder of the coefficients in the companion matrix are 1's in the superdiagonal and are taken care of in the subroutine FCN.

3) The array B contains the coefficients of the output equation and is based on the numerator coefficients of the closed loop transfer function.

4) The last part of the routine contains the loop which calls the IMSL differential equation solving routine DVERK. DVERK solves the set of simultaneous differential equations described in FCN, which uses the coefficients calculated in STIME.

Subprograms called

DVERK (IMSL), FCN (through DVERK)

D.31 SUBROUTINE TABFRQ

This subroutine is the frequency response tabulated data parameter input routine. It creates all of the prompts and menus required to interactively input and change the parameters used to tabulate frequency response data.

Definition of variables

K = Open loop gain
KG = Forward path gain
KH = Feedback path gain
TYPFB = Type of feedback (N=negative, P=positive)
WMIN,WMAX = Range of frequencies for plotting
FMIN,FMAX = Range of frequencies to be tabulated
NOF = Number of frequencies to be tabulated
NLT = Number of lines for tabulation heading
THEAD = Array of heading character strings
TM = Correction routine flag
FD = Output device number

Notes

1) The variables TK, TKG, TKH and TTYPFB are local variables used to hold the associated plotting parameters constant during tabulation. In this way, K, KG, KH and TYPFB (existing common variables) may be used to send information to the routine SCREEN.

2) If frequency response parameters have previously been defined, then reentry is not required. Current parameters are displayed and may be changed, if desired.

3) The corrections routine is used to display, to the user, the parameters just entered. Commanded GO TO statements are used to select the correct line number in the input routine when corrections are necessary.

Subprograms called

FREQR, RNULLI, RNULLR, SCREEN, RCHAR

D.32 SUBROUTINE TABTIM

This subroutine is the time response tabulated data parameter input routine. It creates all of the prompts and menus required to interactively input and change the parameters used to tabulate time response data.

Definition of variables

```
IN = Type-of-input designator
UMAX = Max amplitude of the input
KG = Forward path gain
KH = Feedback path gain
TYPFB = Type of feedback (N=negative, P=positive)
T = Sample period (discrete systems)
TMAX = Max time over which the response is observed
NLT = Number of lines for tabulation heading
THEAD = Array of heading character strings
UIN = Amplitude of the input at time TIM
TINT = Interval between time value in TIME
TIM = Current value of time in TIME
TIME = Array of time values used in calculations
AMP = Array of output amplitude values for each
      time in TIME
TM = Correction routine flag
FD = Output device number
```

Notes

1) The variables TTIN, TTUMAX, TTKG, TTKH, TTYPFB, TTT, and TTTMAX are local variables used to hold the associated plotting parameters constant during tabulation. In this way, IN, UMAX, KG, KH, TYPFB, T and TMAX (existing common variables) may be used to send information to the routine SCREEN.

2) If time response parameters have previously been defined, then reentry is not required. Current parameters are displayed and may be changed, if desired.

3) The corrections routine is used to display, to the user, the parameters just entered. Commanded GO TO statements are used to select the correct line number in the input routine when corrections are necessary.

Subprograms called

FREQR, RNULLI, RNULLR, SCREEN, RCHAR

D.33 SUBROUTINE TEMPLT

This subroutine calculates and plots lines of constant damping ratio and natural frequency for z-plane root locus plots.

Definition of variables

XAF = Length of root locus x-axis
YAF = 10% of length of root locus y-axis
ISTEP = Step factor for interval between lines of constant natural frequency
JSTEP = Step factor for interval between lines of constant damping ratio
P = Array of label values for lines of constant natural frequency
W = Natural frequency
WW = First line of natural frequency to be labelled
S = s-plane coordinate corresponding to some value of natural frequency and damping ratio
RT,IT = Real and imaginary parts of S
Z = z-transform of S
X,Y = Array of real and imaginary parts of Z (plotting points)
D = Damping ratio

Notes

1) The value of S associated with given values of natural frequency and damping ratio is

$$S = -\zeta\omega_n + \sqrt{(\zeta^2 - 1)}\,\omega_n$$

The associated value of Z is

$$Z = \exp(sT)$$

where T is the sample period. The complex values of Z are calculated and plotted by this routine.

2) The density of lines plotted varies depending on the dimensions of the plotting area. The instructions which establish values for ISTEP and JSTEP determine this density.

3) The array P is used so that each line is labelled a max of one time. When a line is labelled, the corresponding element in P is set to zero which, thereafter, indicates that that line should not be labelled again.

Subprograms called

None

D.34 SUBROUTINE TIMER

TIMER is a housekeeping routine called prior to execution of the main time response calculation routines.

Definition of variables

IDOM = Domain of transfer functions
TINT = Time interval
TMAX = Max time over which time response is observed
NOPT = Number of points to be plotted
NOPD = Number of periods between 0 and TMAX (discrete systems)
N1 = Indicates number of periods will be plotted (discrete systems)
AMP = Array of amplitude to be plotted
AMPMIN, AMPMAX = Min and max values in array AMP
PA = Plot or tabulation output flag

Notes

1) The function of this rcutine is as follows:

   a) call CLOOP to calculate loop transfer function

   b) call WINV to w and w' transfer functions to z if necessary

   c) establish the interval of time response calculations based on the domain

   d) call STIME for continuous systems or ZTIME for discrete systems

   e) determine dimensions of plot amplitude axis

   f) call the plotting routine TPLOT

2) A max of 500 points will be plotted for discrete systems. If more than 500 sample periods are covered by TMAK, N1 is increased from 1 as necessary and only every N1 sample period is plotted.

Subprograms called

CLOOP, STIME, TPIOT, WINV, ZTIME

D.35 SUBROUTINE TINPUT

This is the transfer function input and manipulation routine. It creates all of the prompts and menus required to interactively input and change block diagram transfer function parameters. The routine also contains the instructions which multiply together the block transfer functions to form the systems' open loop transfer function.

Definition of variables

NNM = Array containing the order of the numerator of each block
NDN = Array containing the order of the denominator of each block
CONN = Array containing the gain constant of the numerators of each block
COND = Array containing the gain constant of the denominator of each block
BNUM = 2-D array of numerator coeffecients for each block
BDEN = 2-D array of denominator coefficients for each block
RTN = 2-D array of numerator roots for each block
RTD = 2-D array of denominator roots for each block
RR, R1 = Real and imaginary part of a root (input variable)
RN = Array of the numerator roots of the open loop transfer function
RD = Array of the denominator roots of the open loop transfer function
NM = Array of numerator coefficients of the open loop transfer function
NUM = Array of numerator coefficients of the open loop transfer function with leading zeroes
DEN = Array of denominator coefficients of the open loop transfer function
NNUM = Order of the open loop numerator
NDEN = Order of the open loop denominator
A = 1-D array of polynomial coefficients evaluated by ZPOLR
Z = Array of complex roots returned by ZPOLR
NORD = Numerator or denominator operation flag (see notes)
LPNO = Loop number
BN = Block number
NBLKS = Number of blocks
IDOM = Domain of transfer functions
IBF = Intermediate block operation flag (see notes)
IFORM = Form of transfer function
C = Array of coefficients returned by MAKPOL
CC = Array C in reverse order of elements
DIFF = Difference between NNUM and NDEN
CAC = Change and corrections flag (same as TINIT, see DACSAP)
EBN = Expansion block number (see notes)
CCC = Type of polynomial flag (see notes)
M = Parameter change flag
MM = Parameter correction flag
TMINIT, RINIT, FINIT = Parameter initialization flags (see DACSAP)
TFID = Transfer function I.D. number

Notes

1) The block transfer function input routine, starting at line 1010, is broken into three parts:

   a) Up to line 1025 defines the characteristics of the block, i.e., the path in which it exists, the order of the numerator and denominator polynomials and the form of the polynomials.

   b) From 1025 to 1017 is the input routine for transfer functions in coefficient form. This part of the routine also calculates the roots of the blocks input in coefficient form.

   c) Line 1017 to 1154 input transfer functions in factored form.

2) Line numbers 16XX deal with changing transfer functions in another loop. The current loop is saved and the other loop is loaded with these instructions.

3) Line numbers 17XX are used to expand to an outer loop. These instructions save the current loop, calls the routine to form the closed loop transfer functions and places that transfer function in the proper place in the outer loop. Then the outer loop is loaded if it is in a data file or control is sent to the input routine if it is to be entered from the console.

4) IBF is set by the loop change routines and used by the input routine to determine which instructions of the input routine should be executed.

5) EBN describes the block in the outer loop where the inner loop closed loop transfer function is to be placed. This parameter is stored when a loop is saved. When an outer loop is loaded, the closed loop transfer function of the inner loop is automatically loaded into the current block designated by EBN.

6) NORD determines which input instructions should be executed when changes are being made, i.e. numerator instructions, denominator instructions or both.

7) CCC is sent to the SCREEN routine to establish the type of polynomial parameters which should be displayed.

CCC = 1	numerator in coefficient form
CCC = 2	denominator in coefficient form
CCC = 3	numerator in factored form
CCC = 4	denominator in factored form

8) The array TFID contains a number for each block which identifies the block.

TFID = 0	coefficient form in forward path
TFID = 1	coefficient form in feedback path
TFID = 10	factored form in forward path
TFID = 11	factored form in feedback path

Subprograms called

BFILL, CLIN, CLOOP, HELP, MAKPOL, RCHAR, RNULLD,

RNULLI, RNUULR, SCREEN, TLOAD, TSAVE

D.36 SUBROUTINE TLOAD

This subroutine reads data from a data file which has been formed by the subroutine TSAVE.

Definition of variables

II = Type-of-load designator
LPN = Loop number

Notes

1) II determines how much of the data file is read in.

II = 0	Block and system sections are loaded
II = 1	Block section only is loaded
II = 2	System section only is loaded. Block section is loaded into a dummy array.

2) The remainder of the variables used in this routine are defined in TINPUT (section D.35).

Subprograms called

RCHAR

D.37 SUBROUTINE TPLOT

This subroutine contains all of the calls to DISSPLA subroutines necessary to create a time response plot.

Definition of variables

TMAX = Time over which the response is to be observed
AMPMIN,AMPMAX = Min and max values of the array AMP
TIME = Array of time values to be plotted
AMP = Array of amplitude values to be plotted
LINES = Array of plot heading character strings

Notes

1) The plot is rotated on its side if a heading exists. This will cause the plot to better fill the screen and cause any printout of the screen to better fill the page.

Subprograms called

CLIN

D.38 SUBROUTINE TSAVE

This subroutine stores system parameters to a data file under a user specified filename. The routine is called by the user via the DACSAP main option menu or by the program in TINPUT.

Definition of variables

II = Block-section-load designator

Notes

1) Data is saved into two sections. The first section contains either the open or closed loop transfer function of the system. The second contains all other information concerning the system, including the transfer functions of each block.

2) Data in the first section is recalled when individual blocks are loaded from a data file. The second section is recalled when the entire system is loaded.

3) The remainder of the variables used in this routine are defined in TINPUT (section D.35).

Subprograms called

CIIN, CLOOP, RCHAR

D.39 SUBROUTINE WINV

This subroutine performs the inverse transform of w and w' transfer functions to a z-domain transfer function.

Definition of variables

CN = Numerator gain constant
CD = Denominator gain constant
NM = Interim array of numerator coefficients
DN = Interim array of denominator coefficients
CLNN = Order of the closed loop numerator
CLND = Order of the closed loop denominator
CNM = Array of closed loop numerator coefficients
CDEN = Array of closed loop denominator coefficients
Z = Array of roots from subroutine ZPOLR (IMSL)
R = Array of transformed numerator roots
T = Sample period

Notes

1) Given the w (or w') transfer function, G(w),

$$G(w) = \frac{(w + a_1)(w + a_2) \ldots}{(w + b_1)(w + b_2)(w + b_3) \ldots}$$

the following relation was developed for the inverse transform

$$G(z) = \frac{(z + 1)^{m-n} \prod_{i=1}^{n} \{(a_i + C)z + (a_i - C)\}}{\prod_{j=1}^{m} \{(b_j + C)z + (b_j + C)\}}$$

where $C = 1$ for w to z transforms
$C = 2/T$ for w' to z transforms

2) This routine transforms only the closed loop transfer function in preparation for time response calculations (see ZTIME).

Subprograms called

MAKPOL, ZPOLR (IMSL)

D.40 SUBROUTINE WTEMP

This subroutine calculates and plots lines of constant damping ratio and natural frequency for w- and w'-plane root locus plots.

Definition of variables

IDOM = Domain of the transfer functions (w or w')
T = Sample period
XINT = X-interval between lines of constant natural frequency
XMIN = Min value of root locus x-axis
X = Array of x plotting points
Y = Array of y plotting points
XPINT = Interval between values in the array X
WN = Natural frequency
YM = YMAX or ABS(YMIN) whichever is larger
Z = Array of damping ratio values to be plotted

Notes

1) For a given x value and natural frequency

$$y = 2/T \tan \left[T/2 \left\{ WN^2 - \left[2/T \tanh^{-1} (xT/2) \right]^2 \right\}^{1/2} \right]$$

2) For a given y value and damping ratio

$$x = 2/T \tanh \left[\frac{\tan^{-1} (y\ T/2)\ Z}{(1 - Z^2)^{1/2}} \right]$$

3) Five lines of constant natural frequency are plotted at even intervals over the negative x-axis.

4) Lines of constant damping ratio are plotted for Z = .2, .4, .6, .8, .9.

Subprograms called

ATANH

D.41 SUBROUTINE ZTIME

This subroutine calculates the time response of a z-domain transfer function to a step, impulse or ramp input.

Definition of variables

CLNN = Order of the closed loop numerator
CLND = Order of the closed loop denominator
CNM = Array of closed loop numerator coefficients
CDEN = Array of closed loop denominator coefficients
ZNNM = CLNN plus the order of the input numerator
ZNDN = CLND plus the order of the input denominator
QNM = Array of numerator coefficients of the CLTF plus input
QDN = Array of denominator coefficients of the CLTF plus input
A = Local variable used in the long division calculation
AMP = Array of amplitude values to be plotted
TIME = Array of time values to be plotted

Notes

1) The closed loop transfer function is combined with the z-transform of either an impulse, step or ramp function.

2) Long division is used to calculate the output at intervals of the sample period.

Subprograms called

None

BIBLIOGRAPHY

Chen C., Analysis and Synthesis of Linear Control Systems, Pond Wood Press, 1978.

D'Azzo, J.J. and Houpis, C.H., Linear Control System Analysis and Design: Conventional and Modern, McGraw-Hill, 1981.

DiStefano III, J.J., Stubberud, A.R., and Williams, I.J., Feedback and Control Systems, Schaum's Outline Series, McGraw-Hill, 1967.

Dorf, R.C., Modern Control Systems, Addison-Wesley, 1980.

Franklin, G.F. and Powell, J.D., Digital Control of Dynamic Systems, Addison-Wesley, 1980.

Katz, P., Digital Control Using Microprocessors, Prentice-Hall International, 1981.

Lipschutz, S. and Poe, A., Programming with FORTRAN, Schaum's Outline Series, McGraw-Hill, 1978.

Melsa, J.L. and Jones, S.K., Computer Programs for Computational Assistance in the Study of Linear Control Theory, McGraw-Hill, 1973.

Roskam J,. Airplane Flight Dynamics and Automatic Flight Controls, Parts I and II, Roskam Aviation and Engineering Corporation, 1979.

Staff of Research and Education Association, Fogiel M., Director, Problem Solver in Automatic Control Systems/Robotics, Research and Education Association, 1982.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria Va 22314	2
2.	Chairman, Code 67 Department of Aeronautics Naval Postgraduate School Monterey, Ca 93943	1
3.	Library, Code 0142 Naval Postgraduate School Monterey, Ca 93943	2
4.	Dr. Marle D. Hewett H. R. Textron 2485 McCabe Way Irvine, Ca 92714	5
5.	COMO R.H. Shumaker, Code 00 Superintendent Naval Postgraduate School Monterey, Ca 93943	1
6.	LT C.M. Cooksey 4134 Tennyson Ave. Colorado Springs, Co 80910	5
7.	Professor D.J. Collins Code 67Co Department of Aeronautics Naval Postgraduate School Monterey, Ca 93943	2
8.	Distinguished Professor G.J. Thaler Code 62Tr Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, Ca 93943	1
9.	Professor J.H. Duffin Code 62Dn Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, Ca 93943	1
10.	Professor D.E. Kirk Code 62Ki Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, Ca 93943	1
11.	Professor R.D. Strum Code 62St Department of Electrical and Computer Engineering Naval Postgraduate School Monterey, Ca 93943	1
12.	CDR V.C. Gordon 315 Alemeda Blvd. Coronado, Ca 92118	1
13.	LCDR M.S. Kosiek 1915 Wilene Dr. Beavercreek, Oh 45432	1

14. MR Bob Richards
U.S. Naval Test Pilot School
Patuxent River, Md 20670 1

13537 5

211183

Thesis
C74782 Cooksey
c.1 An interactive computer aid for the design and analysis of linear, single input / single output digital and continuous control systems.

Thesis
C74782 Cooksey
c.1 An interactive computer aid for the design and analysis of linear, single input / single output digital and continuous control systems.

thesC74782
An interactive computer aid for the desi
3 2768 002 09392 4
DUDLEY KNOX LIBRARY